UNIVERSITAT POLITÈCNICA DE CATALUNYA (UPC) – BarcelonaTech
UNIVERSITAT DE BARCELONA (UB)
UNIVERSITAT ROVIRA i VIRGILI (URV)

FACULTAT D'INFORMÀTICA DE BARCELONA (FIB)
FACULTAT DE MATEMÀTIQUES (UB)
ESCOLA TÈCNICA SUPERIOR D'ENGINYERIA (URV)

Forecasting Financial Time Series Using Multiple Kernel Learning

Luis Fábregues de los Santos

Master in Artificial Intelligence

Master's thesis

Supervised by Argimiro Arratia* and Lluís Belanche†

To be defended on 5/07/2017

*Dept. of Computer Science, Universitat Politècnica de Catalunya
†Dept. of Computer Science, Universitat Politècnica de Catalunya

Thanks to my tutors Argimiro Arratia and Lluís Belanche for their insight on both Artificial Intelligence techniques and Financial theory. Their patience and proposals kept the project running.

Thanks to my friends Victor Grau and Armando Díaz for their ideas on caching partial results of the algorithms.

Thanks to my family, in particular my parents, for they support during this thesis and the entire master's program.

Finally, thanks to my couple Sandra for cheering me up and being on my side during the days of intense work.

Abstract

This thesis introduces a forecasting procedure based on Multiple Kernel Learning to predict and measure the influence of several economic variables in the process of predicting the equity premium of the S&P 500 Index. In the experiments of Welch and Goyal they determined that, using linear models, those economic variables had an unreliable effect on the predictive capabilities of the models. The experiments performed in this thesis with MKL use the same data in an attempt to predict with non-linear models. The kernels that are part of the MKL procedure are multivariate dynamic kernels adapted for time series. The presented financial variables have a questionable impact on the predictive capabilities of the developed models due to the data being noisy. Some of the kernel methods for time series may not be able to extract any relevant information from exogenous variables, as they are matched in the results by a simple RBF kernel. MKL shows a poor capacity at selecting the best combination of kernels as it is also matched by RBF, and even the kernels that MKL uses. However the experimental results show that the presented methods have better predictive capabilities than the linear models.

Keywords

Support vector machines, Financial time series, Multiple kernel learning, Time series forecasting

Contents

List of Figures

List of Tables

1. Introduction

Financial gain is probably the best incentive to spark research in the fields of mathematics and computer science. The biggest companies are in a perpetual cycle of self improvement, employing a vast amount of resources to make theoretical and practical developments to outplay their business opponents. Organizations providing programming challenges offer enticing prices for the research teams that obtain the best solutions to their problems. Companies that have online presence hire the best researchers in the fields that may improve their internal algorithms. Being money such a successful motivator it is always interesting to research how the economies develop and, if possible, how these developments could be predicted.

Predicting the nature of the market itself has been the objective of much research by the statistics and machine learning groups. In particular the USA stock market returns have been used as data sets for these researches, as the number of recorded information of it is enough to apply learning techniques. The amount of information available does not make the problem easy however. The financial market is volatile, dependant on politics, natural disasters, and business movements.

One of the most representative indicators of the American stock market is the Standard and Poor's Index (S&P Index) and it has been the target of many prediction algorithms. Although the first approaches to this predictive task would only use lagged versions of this variable (just try to predict the S&P Index of tomorrow using the information of its past), soon researchers found new variables to add in to the predictive models. Those variables were shown to increase the capabilities of said models and perform well.

However, the improvement was spurious. Further tests using different data sets or slightly different techniques yielded unsatisfying results. The work performed by Welch and Goyal[40] offers a well-founded answer, concluding that the variables introduced by those other researchers yielded data and model dependent results. This put a slight stop to the creation and promulgation of these kind of variables, and the models using variables that are not the S&P Index.

The current situation holds a somewhat void of improvement in the use of exogenous(not S&P Index) variables. A common criticism to the work of Welch and Goyal is that they made use of mostly linear solvers to test those exogenous variables. On the other hand, newer non-linear approaches to this problem had been developed and tested with some degree of improvement using only the S&P Index. Very complex techniques such as Multiple Kernel Learning[21] have been applied to financial time series yielding good results, with the very interesting feature of providing a weighted vector to each of the input kernels, thus creating a comparison of influence between them in the final model.

There is a gap in the current literature as there are no methods that use complex non-linear solutions to exploit the exogenous variables. Non-linear models may unravel their importance or give another argument against their use. Variety of experimentation is key to this issue so employing several algorithms, methodologies, and data representations is important to give redundancy to the results. Creating a document with said results may help in the present discussion.

The objective of this thesis will be to determine which is the best kernel learning method for these series and to ascertain what is the influence of the exogenous variables on the predictive models. The following work includes:

- A presentation of new kernels for time series with a comparative evaluation of them.
- An experimental procedure that uses the MKL technique to obtain a combined model and weights the composing kernels.
- An empirical comparison between validation techniques for the parameter tuning phase.

This document has the following sections: a State of the Art on which the most recent and insightful works on the field will be commented and analyzed, an introduction to the basics of kernel functions and kernels for time series on the Kernel Theory section, a short review of the kernel learning methods (including MKL) in the Kernel Learning section, an explanation of the data set which the experiments will be based on in the Experimental Data section, the main experimental procedure of the task will be presented in the Classification and Regression Experiment section, and the Exhaustive Experiment will describe more wide-range tests performed on the data.

Part from the work presented in this thesis was summarized in an article and accepted in the International Work Conference on Artificial Neural Networks (IWANN) in spring of 2017. The reference to the published article can be found at `https://link.springer.com/chapter/10.1007%2F978-3-319-59147-6_16`.

2. State of the Art

There is a long history of attempts to predict stock market returns by specifying a regression task using lagged predictor variables independent of the stock market returns. Shiller [34], Campbell and Shiller [7], Cochrane [11], among others, have studied the forecasting of future excess returns using the dividend price ratio as predictor. Other popular predictor variables explored in the literature are the dividend yield, earnings price ratio, dividend-to-earnings ratio, volatility, interest rates, exchange rates, consumption indices and inflation rates (see, e.g., [19], [24], [28], [29] and [13] for a general discussion). The list of valuation ratios sought of as forecasters of expected excess returns is much longer and show *"... a pervasive pattern of predictability across markets wherein the cashflow or price change one may have expected is not what is forecast."* [13]. In view of this and further evidence showing the spurious nature of predictor models (mostly linear regressions on the aforementioned valuation ratios), several authors have conducted extensive studies on the forecasting performance of various economic variables and different models (to mention a few, e.g., [2], [6], [12], [40]). The work by Welch and Goyal [40] is of particular interest since the authors do a comprehensive revision of the empirical performance of the most widely accepted variables as predictors of equity premium, under *linear* regression models, and conclude that these models have poor predictive capacity both in-sample and out-of-sample.

Predicting the market returns has been the object of debate by practitioners given that the data is believed to be non-stationary and with a high degree of noise. The use of the previously mentioned predictor variables has often been a point of controversy given their unreliable results. Among common practises in investment, the "buy-and-hold" strategy is based on buying stocks and holding them without regard of the market. This is further supported by the hypothesis of the efficient market(Fama [18]), which states that the market cannot be predicted and that no excess of return can be obtained by predicting the market.

Regardless of the efficient market hypothesis, many approaches to predict market returns have been developed. In particular, Support Vector Machine approaches with general kernels have been seen in the literature to create such predictive models (see, e.g., [17], [30], [27], [36]). These kernels are, however, not tailored to exploit the time dependency of the data. In this regard, the kernels created in [32] make use of said structures. The work exposed in [21] goes further into the complexity of the algorithms, employing Multiple Kernel Learning as a predictive model.

3. Kernel Theory

The following sections contain a brief description of kernels, the conditions they have to meet to be able to become a proper kernel, how a kernel matrix is defined and created, an introduction to kernels for time series, the many kernels for time series introduced in the literature, and a new kernel developed for this work.

3.1 Kernel basics

Kernels have various definitions depending on the context in which they are used. However, in this thesis, its definition will be the one most frequently used in the statistical machine learning field.

Kernels[25] are two-place symmetric functions that return the inner product of the arguments in some feature space, thereby inducing an implicit mapping that creates an image of the input into the desired feature space. It is then possible to compare input data in a higher-dimensional space without the need of calculating the exact coordinates of the transformation.

These functions are used in the task of dealing with data that cannot be related linearly. By using a kernel as a similarity measure the input data can be projected into a feature space in which it can be related linearly. The projection may be costly to be calculated however there are times in which it is not necessary, the kernel can be calculated using other methods. The technique to evade the actual mapping of samples is known as the 'kernel trick' and it is widely used in machine learning applications.

Kernel functions can be defined as follows:

$$k(x, z) = \langle \phi(x), \phi(z) \rangle \tag{1}$$

where $x, z \in X$ are input vectors. ϕ represents the mapping from the original feature space X into a new feature space F as follows:

$$\phi : x \to \phi(x) \in F \tag{2}$$

3.2 Kernel Matrices and Positive Semi-Definiteness

Using kernel functions it possible to create a kernel matrix [25]. These matrices are particular cases of Gram matrices, which are matrices of inner products of two data samples. In the case of kernel matrices this inner product is substituted by the kernel function. The formulation of the kernel matrix is as follows:

$$K_x = [k(x_i, x_j)]_{i,j \in I} = \begin{bmatrix} k(x_1, x_1) & k(x_1, x_2) & \cdots & k(x_1, x_n) \\ k(x_2, x_1) & k(x_2, x_2) & \cdots & k(x_2, x_n) \\ \vdots & \vdots & \ddots & \vdots \\ k(x_n, x_1) & k(x_n, x_2) & \cdots & k(x_n, x_n) \end{bmatrix} \tag{3}$$

where I is the space of indexes of all the samples.

Kernel matrices serve as the main data structure that will be used by the learning algorithms to create a model. It synthesizes the information of the input data and the information about the features of the data. It also serves as a validation method for the kernel function: any valid kernel function should produce a symmetric and positive semi-definite kernel matrices.

It is known that a function is a valid kernel function if and only if it induces positive semi-definite (p.s.d.) matrices [25]. Said property ensures that methods of convex programming will converge to a global solution using matrices created with those functions. This property is defined as follows:

$$\sum_{i,j=1}^{n} c_i c_j k(x_i, x_j) \geq 0 \tag{4}$$

for all $n \in \mathbb{N}, x_1, ..., x_n \in X$ and $c_1, ..., c_n \in \mathbb{R}$.

This property can be applied directly to kernel matrices defined using a p.s.d. kernel. Any operation applied to a p.s.d. kernel matrix that does not alter that property will generate a viable kernel. Knowing that some basic operations like the sum, product and limit may not change this property (using positive operands), new kernels can be created applying such operations over existing kernels.

3.3 Kernels for time series

Time dependant data sets contain observations obtained with the same frequency. For instance, a data set can have the values of river saline levels measured monthly. This creates a homogeneity on the time stamps of each observation (they will always be separated by one month) but does not assure that a year, for example, will be complete. Failures on the measuring process may occur and the missing observation must be considered in a different way than typical missing data in other problems. The construction of kernels also considers these factors.

Kernels for time series can be constructed using two approaches: structural similarity and model similarity. Structural similarity employs methods to find an alignment of the data that makes possible the comparison between series. Model similarity changes the structure of the data by constructing a higher level representation of it and the comparison is performed using this new representation.

There are different philosophies on building time-dependant kernel functions. Two common approaches are: to build a kernel function around a previously defined model that takes into account the time dependency of the input data or to change the data in order to be used by existing and non-time dependant similarity functions. Both options will be explored on the particular models introduced below.

Identifying the structure of the series can be helpful to find the best method to define predictive models. A deterministic model works under the assumption that the data belongs to a determined function and uses numerical analysis techniques to fill missing values. This undefined function is considered as a combination of polynomial functions [22]. It is the simplest model that can be assumed in a time-series.

Real data sets are rarely deterministic, as the values usually deviate from a linear combination of functions. Models that take into account this consideration are called Stochastic Models. In this kind of model it is considered that future values depend on past values. This fact does not mean that Deterministic Models are useless in time series, but their limitations must be understood.

Determining if a model is deterministic or stochastic often follows an experimental approach: a model of the data is built using linear combinations of polynomial functions and the result is tested against the data. There are also mathematical procedures to determine if a data set follows a deterministic model, like in [4], in which they demonstrate the discriminant capability of a model based in singular value decomposition. The Lag and Autocorrelation plots can also help to distinguish between said models.

Stochastic models are very wide in definition and assume less information about the underlying data. Another assumption must be made in order to apply statistical models and, in the case of Stochastic

Models, that assumption is second order stationarity.

The definition of stationarity states that the distribution of data of an stochastic process is invariant in time. Also, the auto-covariance between X_t and $X_{t+\tau}$ only depends on the lag τ [31]. Given this definition and assigning a value to the lag (which is the number of previous observations that should be taken into account to predict the actual one) it is possible to apply some probabilistic models to the data, assuming that the range in time defined by that variable is relevant.

On the other hand, Non-Stationary time series have their distribution moments change over time. The DickeyFuller test[16] can be used to detect if a data set is Non-Stationary. Another way to determine it is to plot the mean and the variance. If those values change strongly over time, the data might not be Stationary. That property can also be proven by notable changes in auto-covariance or spectrum values [31]. Box plots in particular can be useful to determine this.

Inside of Non-Stationary models, another assumption can be made in order to use statistical methods. A Seasonality model assumes that the properties of the data change through time with defined patterns. Seasonality can be detected by observing repeated patterns in the data, but that can be hard to ascertain. Some experts [26] prefer to fit a model that takes into account seasonality and assess if this characteristic was detected in the data.

Given that the context of the present work is financial time-series, it is generally understood that the data is not stationary, which makes theoretically nonviable the use of the immense majority of methods. A common practise is to work with successive differences of the series in order to make the data resemble a stationary process.

3.4 Vector Auto-Regression Kernel

Vector Auto-Regression(VAR)[20] is an econometric model that relates the data of the observation at point x(t) with a linear combination of lagged values of the observation. Each variable is defined by a function that represents its changes over a defined period of time using past information of itself and other variables. In order to fit a model, a lag parameter is provided, which defines how many time steps the function will be looking at in the past to assess the linear combination parameters. Vector Auto-Regression is a model by itself, capable of predicting values of new samples, but the information generated by it can also be used to create a kernel.

Vector Auto-Regression kernels can be built using three steps:

1. Build two VAR models with two series and fit them using a certain number of lags.

2. Append the values of the transition matrices and intercepts of each series and calculate the Frobenius norm over the difference of those values.

3. Apply the Radial Basis Function to the Frobenius distance to convert it to a similarity measure.

The VAR model that relates the data of the observation at point $x(t)$ is the result of a linear combination of all the variables of the observation. The formulation is as follows:

$$x(t) = \sum_{l=1}^{L} A_l x(t-l) + b + \varepsilon_t \tag{5}$$

where $x(t)$ is the sample at time t, L is the number of lags of the model, A is the transition matrix (a square matrix with the same dimensions as features has the data), b is the intercept (a vector of the dimension equal to the number of features) and ε_t is the Gaussian noise at time t.

The VAR function can be used to build a model similarity kernel, as seen in [32]. In order to compare VAR models it is interesting to consider the transition matrices and the intercept vectors. A simple matrix can be built appending the intercept as an additional column of the transition. This results in $\hat{B} = (A_1|A_2|...|A_L|[b])$. In order to calculate the difference between series s_1 and s_2 the difference between $\hat{B}_{s_1}$ and $\hat{B}_{s_2}$ is calculated and then the Frobenius norm is applied:

$$FD(s_1, s_2) = \sqrt{Trace\left\{(\hat{B}_{s_1} - \hat{B}_{s_2})(\hat{B}_{s_1} - \hat{B}_{s_2})^T\right\}} \tag{6}$$

Once this Frobenius distance is calculated, the distance can be transformed into a valid kernel using the Radial Basis Function(RBF) kernel:

$$k_{VAR} = exp\left\{\frac{-FD(s_1, s_2)}{2\sigma}\right\} \tag{7}$$

The parameters of this kernel methodology are the number of lags L and σ. The value of L will be fixed to five. σ will be set to the median Frobenius distance between the time series being compared. Both these parameters are set following the indications of [32].

3.5 Global Alignment Kernel

Global Alignment(GA) is a generalization of a well-known family of distance and similarity measures called Multivariate Dynamic Time Warping(MDTW) introduced in [33]. In order to understand Global Alignment it is necessary to explain how MDTW works in the context of time series and which are its drawbacks.

The objective of Dynamic Time Warping is to measure the distance between two series. In order to do so, both series should be aligned. The core of the problem is to determine the best alignment between the two series and, using that alignment, measure their similarity.

An alignment in MDTW is represented by a set of relationships between a point in the series and another point of the same series or the other one. Considering s_1 and s_2 as two time series, those relationships are the following: $s_1(t)$ with $s_2(t)$ denoted by $\rightarrow$, $s_1(t)$ with $s_1(t+1)$ denoted by $\uparrow$ and $s_1(t)$ with $s_2(t+1)$ denoted by $\nearrow$.

The relationships are represented as two integer vectors π_1, π_2 of the same length with binary increases. Each item of the vectors is a relationship between elements of the series. The length of the vectors is always equal to or less than the length of the smallest series. Each of the previously mentioned relationships can be represented on those vectors as follows: $(0, 1)$ for $\rightarrow$, $(1, 0)$ for $\uparrow$ and $(1, 1)$ for $\nearrow$. Intuitively, each vector $\pi_1(t)$ indicates an element of s_1 that forms a relationship with the element $\pi_2(t)$ of s_2. For the sake of simplicity the two vectors that represent the alignment will be denoted as π. Those alignments, by definition, only consider values of zero or one lag in both series.

After obtaining a satisfying alignment, the distance between the series can be obtained as follows:

$$D_\pi = \sum_{i=1}^{|\pi|} d(x_{\pi_1(i)}, y_{\pi_2(i)}) \tag{8}$$

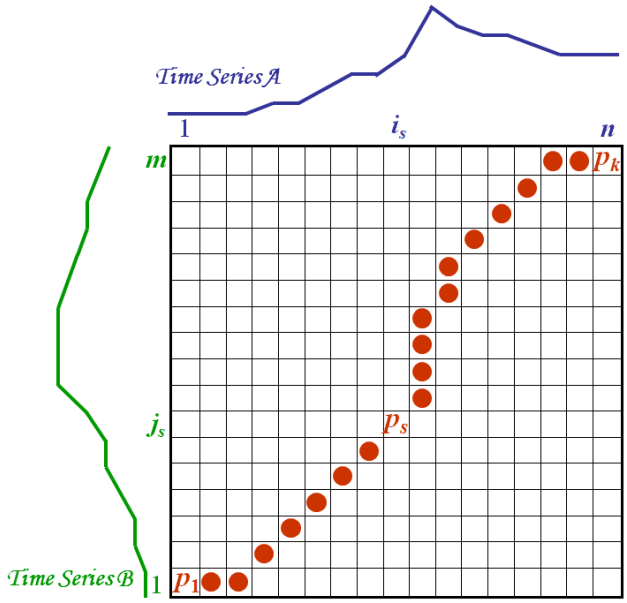

Figure 1: Multivariate Dynamic Time Warping. Source: [37]

where the distance function d can be any metric, most commonly the Euclidean distance.

The presented algorithm is capable of finding more than one alignment. The selected alignment will be the one that minimizes the distance between the series. The formulation of the final MDTW distance is as follows:

$$MDTW(s_1, s_2) = \frac{1}{|\pi^*|} \min_{\pi \in A(s_1,s_2)} D_\pi(s_1, s_2) \tag{9}$$

where $|\pi^*|$ is the length of the alignment with less distance and A is the set that contains all possible alignments π.

This distance measurement does not fulfill the positive semi-definite requirement to form a kernel, even after applying the Radial Basis Function. For this reason the Global Alignment, explained in [15], generalization can be more widely applied, which delivers correct kernels and enables the creation of a structural similarity kernel.

Global Alignment follows the same computational steps as MDTW. However, instead of selecting the alignment with minimum distance, it considers all the alignments. This makes kernels defined by this metric positive semi-definite under mild assumptions. This is based on the notion that all alignments provide information about the similarities between both series.

The formulation of this kernel function can be expressed using several distance metrics. The following formula is the one used in the context of the thesis:

$$k_{GA}(s_1, s_2) = \sum_{\pi \in A(s_1,s_2)} e^{-D_\pi(s_1,s_2)} = \sum_{\pi \in A(s_1,s_2)} \prod_{i=1}^{|\pi|} k(x_{\pi_1(i)} \cdot y_{\pi_2(i)}) \tag{10}$$

where,

$$k(s_1, s_2) = \frac{\frac{1}{2}exp(-\frac{1}{\sigma^2}\|x-y\|^2)}{1-\frac{1}{2}exp(-\frac{1}{\sigma^2}\|x-y\|^2)} \tag{11}$$

where x and y are elements of s_1 and s_2 and σ is a parameter which is obtained from the adaptive grid: $\sigma \in \{0.2, 0.4, ..., 2\} \cdot median(\|x(t_1) - y(t_2\|) \cdot \sqrt{median(|x(t_1|)}$, where $x(t_1)$ and $y(t_2)$ are samples in the series. In [32] they use a rank-based approach to obtain that parameter. They retrieve the pair of observations at which the output variable variation reaches minimum in the series. This is an adapted formulation from the original proposed by [15] in which the points are chosen randomly.

An improvement over GA was introduced with the name of Fast Global Alignment or Triangular Global Alignment. This improved version aims toward reducing the computational time of the procedure. That is accomplished by using an extra parameter T that restricts the number of alignments taken into account during the final calculation of the k_{GA}. In particular, lower values of T make the kernel function use alignments close to the diagonal. Increasing the value of T increases the range of alignments that are taken into account.

3.6 Multivariate Dynamic Euclidean Distance Kernel

In the same line as Global Alignment Kernel the Multivariate Dynamic Euclidean Distance Kernel (MDED), introduced in [32] is a structural similarity model that creates an alignment of data between two series of different size in order to be able to compute the distance measure. MDED opts for a much simpler approach, as it removes the first elements of the longest series until it matches the size of the shortest time series.

Even if this alignment is potentially worse in most of the cases with respect to MDTW, MDED is computationally less expensive. The approach is also backed by financial theory: observations generated in later time stamps contain information from older ones.

Having that $L1 \leqslant L2$ where $L1$ and $L2$ are the lengths of vectors s_1 and s_2 respectively, this alignment is defined in the notation of MDTW as $\pi_1 = [0, 1, 2, ..., L1-1, L1]$ and $\pi_2 = [L2-(L1-1), L2-(L1-2), ..., L2-1, L2]$. Using said alignment, the calculation of the distance between the series is done as in eq. 8. Again, the metric employed is the Euclidean distance. In a similar line to the k_{VAR} calculation, it is necessary to calculate the RBF kernel using the defined dissimilarity measure in order to obtain a p.s.d. kernel:

$$k_{MDED} = exp\left\{\frac{-D_\pi(s_1, s_2)}{2\sigma}\right\} \tag{12}$$

The parameter of this kernel function, σ is estimated using the median of all D_π of the available data, as suggested in the work that introduces the algorithm [32].

3.7 Multivariate Dynamic Arc-Cosine Kernel

Arc-Cosine kernels have interesting properties[10]. Their behaviour is similar to a neural network with one infinite hidden layer. This kernel function can be defined using different degrees that have different properties with slim variations in formulation. In the authors define most of this properties and make an

experimental comparison of Arc-Cosine kernels with Radial Basis Functions. They obtain good results on challenging data sets, surpassing other SVM, and comparable results with deep belief networks.

The basic formulation of the Arc-Cosine kernel function depends on the angle between the samples. The angle between samples can be defined with the following formulation:

$$\theta = cos^{-1}\left(\frac{s_1^T s_2}{\|s_1\|\|s_2\|}\right) \tag{13}$$

The formulation of the function is defined by the degree n, which regulates its complexity. The kernel function can be expressed as follows:

$$k_n(s_1, s_2) = \frac{1}{\pi}\|s_1\|^n\|s_2\|^n J_n(\theta) \tag{14}$$

J_n is a family of functions that analyze the complex dependencies of the angle. The formulation is quite complex for an arbitrary value of n. However, in the context of this thesis, only $n = 0$ and $n = 1$ will be considered. The different formulations for both this degrees are:

$$J_0(\theta) = \pi - \theta \tag{15}$$

$$J_1(\theta) = sin\theta + (\pi - \theta)cos\theta \tag{16}$$

Arc-Cosine kernels have different properties depending on the degree of the formulation, with many complex implications that can be found in the referenced work. This kernel function only has as parameter the degree, n, which has a small window of tuning. It makes that this kernel function has potentially bad results compared to other kernel functions that allow some parametrisation.

Arc-Cosine kernels as defined in [10] and, in the same line as the Euclidean Distance Kernel, are created to work with complete data. In order to work with time series an alignment must be used to obtain input data for the formula, thus creating a structural similarity model. The chosen alignment is the one presented in 3.6 for its simplicity and re-usability.

4. Kernel Learning

After determining and forming the appropriate kernel matrix using the functions described in the previous section, a model is built to fit to the data and provide predictive capabilities. Kernel learning methods provide a process to create such models and tools to use them for prediction tasks.

The next sections will describe both general and particular kernel learning models: first a brief introduction on building Support Vector Machines for single kernels functions; second a conceptual description of Multiple Kernel Learning; and finally two particular MKL algorithms that will be employed in the tests.

4.1 Support Vector Machines

Support Vector Machines(SVM)[38] are predictive models. The learning algorithm in the case of classification is designed to create the biggest separation possible between samples of different classes. This is accomplished by creating a separating hyper-plane that divides the samples in two groups. The distance between this hyper-plane and the closest samples of each class is called margin and the algorithm maximizes it. The samples that are in the margin are called support vectors and they define the shape of the hyper-plane. Similarly the regression techniques tries to create a hyper-plane that has the minimum distance to the samples as possible, considering a margin for errors.

The simplest formulation of this technique for classification is the following:

$$\min \frac{1}{2} w^T w \tag{17}$$

subject to

$$y_i(w^T x_i + b) \geqslant 1, i = 1, ..., m \tag{18}$$

where w is the normal vector to the hyper-plane, b is the offset of the hyper-plane from the origin, x are the training samples and y are the training tags. The formulation of the regression problem[35] only changes the conditions of the equation of the classification problem:

$$\begin{aligned} y_i - w^T x - b &\leqslant \varepsilon \\ w^T x + b - y_i &\leqslant \varepsilon \end{aligned} \tag{19}$$

where ε determines the maximum deviation from the function defined by $f(x) = w^t x + b$ to the samples. This formulation, however, does not allow non-linearly comparable data to be related correctly and the model is not influenced by the user with any parameter. In order to address those features, ν-SVM[8] can be used for the classification problem:

$$\min \frac{1}{2} w^T w - \nu\rho + \frac{1}{m} \sum_{i=1}^{m} \xi_i \tag{20}$$

subject to

$$\begin{aligned} y_i(w^T \phi(x_i) + b) &\geqslant \rho - \xi_i, i = 1, ..., m \\ \xi_i &\geqslant 0, \rho \geqslant 0 \end{aligned} \tag{21}$$

where ρ is a free parameter that serves as threshold, ξ_i are the residual errors and ϕ is a function that maps the data to a higher dimensional space (a kernel in the context of this thesis). A similar formulation can be applied to the regression problem[9]:

$$\min \frac{1}{2} w^T w + C \left(\nu\varepsilon + \frac{1}{m} \sum_{i=1}^{m} (\xi_i + \xi_i^*) \right) \tag{22}$$

subject to

$$\begin{aligned} &(w^T \phi(x_i) + b) - y_i \leqslant \varepsilon + \xi_i, \\ &y_i - (w^T \phi(x_i) + b) \leqslant \varepsilon + \xi_i^* \\ &\xi_i, \xi_i^* \geqslant 0, i = 1, ..., m, \varepsilon \geqslant 0. \end{aligned} \tag{23}$$

where C is the regularization parameter. ν is an interesting parameter in both problems. It serves both as an upper bound for the margin errors and a lower bound for the number of support vectors with respect to the number of training samples. Different values of this parameter can make the model behave in substantially different ways so its values should be chosen carefully.

4.2 Multiple Kernel Learning

Multiple Kernel Learning(MKL)[23] is a research field that aims to find the best combination of kernels to solve a task. It is possible for a problem to have several kernel functions that cover different characteristics of the data, or different representations of the same data. Those procedures create different kernel matrices that can be used to train a predictive model. However, it is also possible to combine the information of those matrices into a single combined matrix of kernels. MKL makes it possible to combine in the same predictive model information obtained using different techniques.

The mathematical formulation of this process is the following:

$$k_\eta(x_i, x_j) = f_\eta(\{k_m(x_i^m, x_j^m)\}_{m=1}^{P} | \eta), \tag{24}$$

where k_η represents the combined kernel, f_η is the combination (linear or non-linear) function, k_m represents a kernel function for a set of P kernels and η parametrizes the combination function. This particular formulation is for the case of the parameters of the combination function being fixed.

The combination functions of MKL procedures often obtain a vector of weights and perform a linear combination of the kernel matrices or functions weighted by the obtained vector. In order to infer the vector of weights, several approaches can be used. One of the most employed in practise is the optimization procedure by which the kernel weights are obtained using an optimization algorithm following a certain criteria. In the case of the convex sum of weights (all weights must be positive and sum one) the general optimization procedure is the following:

$$\arg\max_{\omega} A(\omega k, Y), \tag{25}$$

where ω is the vector of weights, k is the list of input kernel functions, Y is the vector of response values and A is a similarity measure.

Two algorithms to perform this task are employed in the development of this work. EasyMKL, a classification approach of MKL based on the probability distribution of each class, and Center Alignment, a MKL algorithm for regression based on similarity measures between kernel matrices.

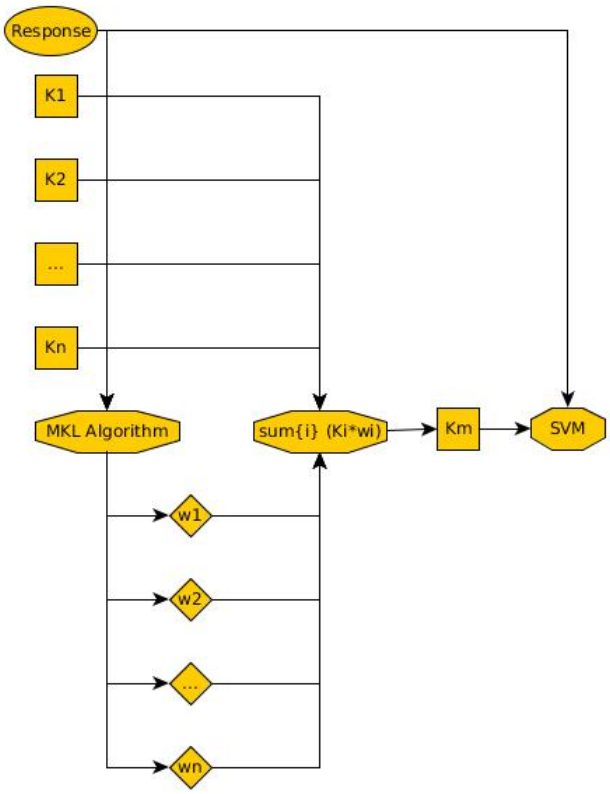

Figure 2: Multiple Kernel Learning general procedure. *Ki* are kernel matrices, *wi* are the weights computed by the algorithm, and *Km* is the combined kernel matrix.

4.3 EasyMKL

The EasyMKL[1] algorithm obtains the parameters of the combination function using an optimization approach, in particular solves a max-min problem involving the parameters of the combination function η and the probability distribution of each class γ. After the η weights are obtained they are combined convexly (the optimization restrictions ensure that the weights are positive and sum one). The l-1 norm is used as a structural risk function to guide the optimization. As base learner it uses Kernel Optimization of the Margin Distribution(KOMD), a kernel classifier that performs direct optimization of the margin distribution.

The formulation of this approach starts with defining the convex combination:

$$k_\eta = \sum_{m=1}^{P} \eta_m k_m, 0 \leqslant \eta_m \leqslant 1, \tag{26}$$

where, in this case, η_m is the assigned weight to each kernel matrix. The initial optimization equation is:

$$\max_{\eta:\|\eta\|=1} \min_{\gamma\in\Gamma} Q(\eta,\gamma) = \max_{\eta:\|\eta\|=1} \min_{\gamma\in\Gamma} (1-\lambda)\gamma^T \hat{y}(\sum_{m}^{P} \eta_m \widehat{k_m})\ \hat{y}\gamma + \lambda \|\gamma\|^2 \tag{27}$$

where λ is an exogenous parameter of the optimization process, $\hat{y}$ is the vector of training set classes and $\widehat{k_m}$ is a kernel matrix of the training set. Using $d_\eta(\gamma)$, which is a vector containing all $d_m(\gamma) = \gamma^T \hat{y} \widehat{k_m} \hat{y}\gamma$,

the formulation of the problem can be simplified to:

$$\min_{\gamma \in \Gamma} (1 - \lambda) \tilde{\eta^*} d_\eta(\gamma) + \lambda \|\gamma\|_2^2 \tag{28}$$

where,

$$\tilde{\eta^*} = \frac{d_\eta(\gamma) \|d_\eta(\gamma)\|_1}{\|d_\eta(\gamma)\|_2 \|d_\eta(\gamma)\|_2} \tag{29}$$

The implementation of this algorithm is quite simple, as a optimization MKL algorithm. The function to minimize is 28 subject to several constraints presented in the introduction of this technique. The authors also provide a pseudo-code for a greedy version of the algorithm in one of the appendixes for better comprehension.

As commented previously, the base learner is KOMD. That classification method is the internal procedure that the EasyMKL method uses for obtaining results and compute the optimized parameters. Common implementations of this method include the capability of predicting values using the KOMD classifier. However, in the context of this thesis, the best procedure is using the parameters calculated by EasyMKL, calculating the combined kernel matrix, and using that matrix to train a Support Vector Machine. This makes the built system capable of more accurate predictions as the ν parameter is of capital importance for these kernels.

4.4 Center Alignment MKL

This algorithm performs kernel alignment using centered alignment[14], which is a similarity measure that can be used to compare kernel matrices. The first step in this procedure is to obtain centered kernel matrices, which are defined as each kernel matrix itself minus the expected value of the kernel function. Centering the features is a process that also centers the resulting kernel matrix which, in turn, improves the performance of the algorithm. Centering also solves previous problems with unbalanced data sets. This process effectively centers the feature mappings in the kernel. This transformation can be defined as:

$$k_c = \left[I - \frac{11^T}{F} \right] k \left[I - \frac{11^T}{F} \right], \tag{30}$$

where I is the identity matrix, F is the number of features, 1 is a vector of ones of $1xF$ elements. k_c is positive semi-definite as it is defined as an inner product.

The MKL algorithm can be constructed using this formulation. Weights are calculated by maximizing the alignment between the combined kernel matrix and the target kernel matrix. The resulting weights can be calculated using simple quadratic programming optimization techniques. This particular formulation is for the convex version of the process:

$$\tilde{\eta^*} = \underset{\eta \in \tilde{\eta}}{argmax} \frac{\eta^T a a^T \eta}{\eta^T M \eta}, \tag{31}$$

where

$$a = (\langle k_{1c}, yy^T \rangle_F, ..., \langle k_{Pc}, yy^T \rangle_F) \tag{32}$$

where $\tilde{\eta}$ is the set of possible weights with convex restrictions, a is a vector which represents the Frobenius distance between each centered kernel matrix to the response kernel matrix, M is a symmetric

matrix that contains the Frobenius distances for each combination of kernel matrices k_{jc} and k_{lc} in $j, l \in \{1, P\}$. The implementation of this algorithm is also straight forward: equation 31 is introduced in a quadratic programming optimization algorithm with the problem constraints presented in the referenced work, obtaining the weights in the process.

5. Experimental Data

The experiments in this thesis use the same data as in [40] and are performed with a similar set of variables. It is a data set that comprises several financial features measured monthly, quarterly and yearly in the range of years between 1871 and 2005. It includes information from several sources, which will be commented besides the explanation of the features.

The objective of the work presented in [40] is to determine if several variables deemed by several researchers as predictors of financial returns have any real impact in general circumstances. The authors perform their experiment using mostly linear regression and comparing the impact of such variables under those models. Their objective is to compare all the variables fairly and determine which of them have an impact on the prediction of S&P 500 equity premium. They determine that all of them have questionable impact.

The objective of this work is to use a subset of those variables described in [40] with non-linear models. The data set offered by the authors of [40] contains many features, many more that is computationally possible to compare in the time frame for this thesis. For this reason, a selected set of features (either extracted or derived from the original data) will be employed in this work. What follows is the list of the terminology used for the target value and features, and a short description comparing them.

Target value

Equity Premium: A representation of the stock market returns, in this case of the S&P 500. It is calculated combining several features from the original data $ep_t = log((Index_t + D12_t)/Index_{t-1}) - log(Rfree_t + 1)$ where *Rfree* is the risk-free rate. This feature is theoretically the return rate of an investment with zero risk, however in practice it is obtained from the interest rate of a three month U.S. Treasury bill. The variables associated with *Index* and $D12$ are the S&P 500 index and the dividends, respectively, and will be discussed in later sections.

Features

Stock Returns: The original problem uses S&P 500 index (also mentioned as SPX or simply *Index*) returns obtained from Center for Research in Security Press (CRSP) and the website of Robert Shiller. This variable encapsulates the market capitalizations of the largest public companies and serves as an indicator of the U.S. economy. In the context of this problem, SPX will be considered the endogenous variable.

Dividend Price Ratio: Both this feature and the following are dependent on the dividends($D12$), which are a moving sum with a window of 12 months of the dividends paid on the S&P 500 index. The dividends data is obtained from the Shiller's website and the S&P Corporation. The formula for the Dividend Price Ratio is $dp_t = log(D12_t) - log(Index_t)$.

Dividend Yield: Very similar to Dividend Price, but Dividend Yield considers past values of SPX $dy_t = log(D12_t) - log(Index_{t-1})$.

Earning Price: In a similar line to the features created using dividends, earning price($E12$) is the moving sum of earnings from S&P 500 index in a window of 12 months. Part of this data is extracted from Shiller's website and the other part is the result from an interpolation process by the authors. The Earning Price is formulated as $ep = log(E12_t) - log(Index_t)$.

Stock Variance: It is a variable that encapsulates the sum of daily returns of SPX. This data was obtained by the authors with the help of G. William Schwert and CRSP.

Book-to-market ratio: It is the ratio of book value to market value for the Dow Jones Industrial Average. For the months from March to December, this is computed by dividing book value at the end of the previous year by the price at the end of the current month. For the months of January and February, this is computed by dividing book value at the end of two years ago by the price at the end of the current month. Book values from 1920 to 2005 come from Value Lines website.

6. Classification and Regression Experiment

As stated in the introduction, the objective of this work is to re-examine the experiments performed by Goyal and Welch in [40] with kernel functions and using multiple kernel learning to weight each feature in order to determine its relative influence in the prediction of the equity premium of SPX.

The experiment is divided in two tasks: regression and classification. The objective of the regression problem will be to predict as accurately as possible the equity premium of the next month. The classification task uses the sign of the equity premium of the next month instead. Positive values of this metric indicate good preconditions to hold a share and negative values indicate a proper time to sell those shares. The reason behind performing both these tasks is to provide a robust response to the usability of the proposed features.

The experimental process follows two phases. In the first phase only endogenous variables will be used. All the kernel methods will be evaluated individually and a Multiple Kernel Learning model will be built using those kernel functions as well. The objective of this first phase is to determine which method performs better in the prediction task. The second phase uses the exogenous variables over the best model of phase one. For each variable, a data frame is built and its influence on the result is observed using MKL. Different combinations of variables and lags will be used. All these experiments will be explained in the following sections.

Data Preparation

Kernel methods for time series do not work with the same data structures as general ones. Kernel functions as VAR or GA require data structures that are comprised of a window of observations in order to extract temporal relationships between them. Those data structures in this context will be blocks of a given size composed of sequential observations. The data compression process transforms the raw original financial information into several blocks of data. Knowing that the observation frequency of the data is monthly, an intuitive way of creating blocks is to build yearly data structures containing 12 observations each. Each of these blocks will be considered as a data point inside of the problem. Kernel matrices will be built applying the different kernel functions to each pair of data points inside of a set.

The sets of data points can be defined in several ways. The classical approach is to divide the available, labeled data in to three sets: a set to train the model, a set to validate the parameters and a set to test the model. Although this approach is very common and applicable to most problems, time series behaves in a different way. For instance, a common method to obtain these sets is using N-Fold Cross-Validation. In a time dependent problem this methodology allows to test a model with a set of data older than the data used to train the model, which is paradoxical. Leave-N-Out solves partially this problem but introduces a new one, which is the time difference between the set of data used to train the model and the set destined to testing. Trying to predict a result in the current year using data from seven years ago may result in misleading performance. Time dependent models are built considering that they should be evaluated with data of the near future. In order to reduce these effects a moving window approach will be employed.

In this thesis the moving window is defined as a set of data points with a fixed number of elements or years. After a window's data is used to train, validate, and test a model the next window will start one year later with the same number of elements. This window will be divided inside the problem in train, validation and test sets. In order to respect the sequential nature of the problem only the last data point will be used as test data. Validation and train sets will contain a preselected number of samples, 2 and 20

respectively. The size of the window in this case is 23 (the sum of the elements of the three sets).

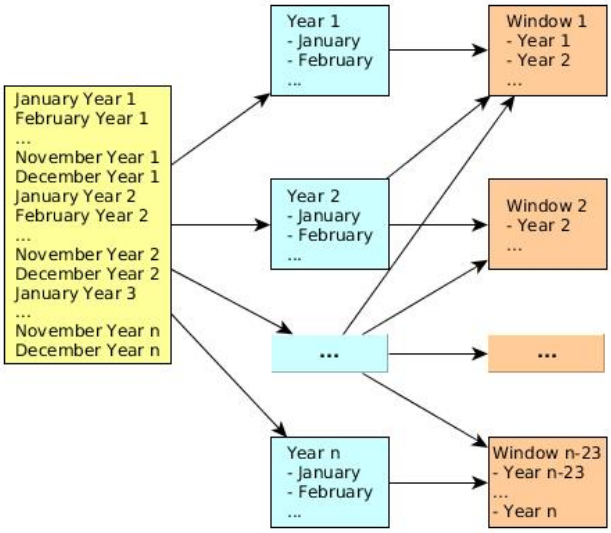

Figure 3: Schema of the data preparation for this task. The first object(left) contains all monthly observations which are divided in yearly blocks of data(middle) to be part of windows of data points(right).

The output variable of this process varies depending on the task that is being evaluated. As introduced, the equity premium is the value to predict in the task of regression. The classification task uses only the sign to create a binary problem. The vector of the true output variables is computed before starting any model building and uses the value of the first month of the next year.

Working with time dependant models introduces an important dependency with past observations of the same variable. It is a common practise to include several lagged observations to increase the information provided to the model and form better temporal models. This practise will be applied to all the predictor variables, with different configurations and quantities of lags.

Performance Metrics

Metrics are strongly related to the type of task being performed. The results of a regression task are real values and they must be compared with the true value. In this case, the model will predict the equity premium of the following month and it will be compared with the true value. Mean Squared Error is a very common performance metric defined as:

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(\hat{y}_i - y_i)^2 \tag{33}$$

Three different errors will be taken into account in the results: train error, validation error and test error. Test error measures the performance of the models against data out of sample, train error does the same for the in sample data and validation error helps to determine if the validation procedure is working correctly.

The performance metric selected for the classification task is the accuracy, defined as the number of correctly classified samples over the total number of samples. This simple metric is selected because it is

intuitive to use in classification problems and it is easy to compare models using it. This metric will also be applied to training, validation and test predictions.

The other metric employed to display results is the set of weights of the multiple kernel learning model. As stated in the definition of the work the weights of a multiple kernel learning can be helpful to determine the relative importance of each kernel. This metric will help to determine which kernel function creates the best models from the data and which kernel matrix, constructed with the different variables, has a higher impact on the prediction of the results.

Validation methods

The process of parameter tuning introduces a new data division problem inside each window, but with several changes with respect to the main approach. Having a concealed, small dataset for validation purposes results in destining only a few samples to predicting and obtaining performance metric for each parameter combination. Many implementations apply out-of-sample as a simple solution however there are arguments to also use K-Fold Cross-Validation. In this thesis, results produced by both will be reported. Although in past sections both these techniques have been discouraged, the problem to solve is not the same. In this section the algorithms will be used to find the best set of parameters for the final model so, using for instance Cross-Validation, does not build a model with future data, only uses it to find the set of model parameters.

The authors of [3] argue that the use of K-Fold Cross-Validation(CV) is possible under several circumstances. In their work they state that this methodology is not used by many practitioners because it employs future data. The use of not-known data during the evaluation process can change the distribution of the resulting model and also alters the order of the observations when the folds are rearranged, eliminating natural relationships. However commonly used methods, as out-of-sample, do not use the available data in an efficient way. The research work performed in [3] suggest that the common effects of using CV are negligible under some conditions. Those conditions are:

- Independence between signals
- Stationarity of the distribution of the data
- The data must be used in an auto-regressive model

The authors also comment that their demonstration is applicable to other lagged non-parametric models. They demonstrate those claims by performing several experiments using data with different underlying distributions. The models increase their predictive capability using CV when the data is generated by an auto-regressive model. However, if the data is seasonal, CV performs worst that the rest of the models. Given the assumptions about the data that this thesis assumes it is only fitting to include both validation procedures and observe how they compare.

Methodology

The problem's methodology is divided in two steps. The first is to determine which is the best single or combination of kernel functions to perform the described predictive task. The second is to use the selected method to determine the relative importance of each exogenous variable using the weights of the trained Multiple Kernel Learning model.

The first step follows the classical methodology of model comparison: perform a predictive task using the same data and compare the results using a predefined metric. The kernel functions defined in 3 will be used training a ν-SVR or a MKL model. The results of each experiment will be shown in a comparative table. The process includes parameter tuning for each of the models. In the case of the single kernel functions, the creation and evaluation of the model is straight forward: a support vector machine is built and fitted using the training data and tested or validated with the rest of the data. In the case of Multiple Kernel Learning, all kernel functions introduced in 3 will be used to create several Gram Matrices that will become the inputs for the MKL procedure. The implementation of KernelAlign for regression lacks of a built-in method to perform predictions, for this reason the resulting weights will be used to create a combined matrix that will be the input to a ν-SVR, which will become the predictive model. EasyMKL has an internal predictor, however the results are rarely good, as it lacks many of the parameters that Support Vector Machine permits. In a similar manner to the regression tasks, the combined kernel matrix can be feed to a ν-SVM and this methodology increases the predictive capability of the model. The method selected in this step will be considered as the internal method.

The second step tries to determine the relative importance of each exogenous variable by using the weights of MKL, the external method. The MKL must be feed with a list of matrices, each one containing a different variable combination. To create such matrices the procedure defines data frames, which are groups of yearly data containing different features. The input list to the problem contains one data frame for each of the five exogenous variables, one data frame with only the endogenous variable and one data frame with all the variables. The data frames containing exogenous variables also contain a certain number of lags of the endogenous variable. The result of the MKL procedure will generate a weight for each data frame, thus weighting the relative importance of each variable. Each data frame will contain the training and validation data for each internal method, creating a kernel matrix that will be included in the list of inputs to MKL. After the computation, the performance of the model can be measured using the predefined performance metric for each task and the distribution of weights.

Using this representation, the input of each experiment will be a list containing seven matrices encapsulating the following features:

1. SPX
2. SPX and Dividend-Price Ratio
3. SPX and Earning Price
4. SPX and Dividend Yield
5. SPX and Stock Variance
6. SPX and Book-to-Market Ratio
7. All the features

The experiments will be constructed including lagged versions of these features as additional information for the modeling process. Four experiments will be defined containing the seven data frames previously commented. Each experiment will be denoted by EX.

- EX1 only considers the exogenous variables without lags and SPX with four lags is included in each data frame.

	SPX	DP	EP	DY	SVAR	BM
EX1	0:4					
	0:4	0				
	0:4		0			
	0:4			0		
	0:4				0	
	0:4					0
	0:4	0	0	0	0	0
EX2	0:4					
	0:4	0, 3				
	0:4		0, 3			
	0:4			0, 3		
	0:4				0, 3	
	0:4					0, 3
	0:4	0, 3	0, 3	0, 3	0, 3	0, 3
EX3	0:4					
	0:4	0:4				
	0:4		0:4			
	0:4			0:4		
	0:4				0:4	
	0:4					0:4
	0:4	0:4	0:4	0:4	0:4	0:4
EX4	0:4					
	0	0:4				
	0		0:4			
	0			0:4		
	0				0:4	
	0					0:4
	0:4	0:4	0:4	0:4	0:4	0:4

Table 1: The definition of the four experiments with their respective lags.

- EX2 adds to the information of EX1 each exogenous variable with a lag of 3; this is motivated by the fact that the resulting data frame will contain more information but without adding too much redundancy.
- EX3 includes four lags of each exogenous variable.
- EX4 contains the same information as EX3 for the exogenous variables, but only includes the SPX without lags.

A visual representation of this data can be seen in table 1.

6.1 Classification task

Tables 2 and 3 show the evaluation of the different kernel methods and multiple kernel learning using EasyMKL. These results are obtained using the same data for each method but adjusting the parameters

individually. The parameter ranges used in each of these methods can be found in A. All the methods are also include both parameter validation methods for comparison.

	k_{VAR}	k_{GA}	k_{MDED}	k_{MDARC0}	k_{MDARC1}	MKL	MKL Norm
Train Acc.	0.588	0.739	0.722	0.674	0.734	0.982	0.777
Validation Acc.	0.872	0.859	0.74	0.491	0.452	0.529	0.763
Test Acc.	0.631	0.64	0.64	0.64	0.658	0.64	0.694

Table 2: Results of kernel combinations using Out-Of-Sample Validation.

	k_{VAR}	k_{GA}	k_{MDED}	k_{MDARC0}	k_{MDARC1}	MKL	MKL Norm
Train Acc.	0.61	0.762	0.742	0.675	0.729	1.000	0.898
Validation Acc.	0.751	0.691	0.674	0.559	0.445	0.417	0.568
Test Acc.	0.568	0.649	0.631	0.640	0.640	0.667	0.658

Table 3: Results of kernel combinations using Cross-Validation.

The results of this table have several interesting factors to cover. All of the test accuracy measures are in very defined range, between the 55% and 70%, which indicates the general capacity of Multivariate Dynamic kernels for this task. Individual kernels share similar test accuracy measures, around the 64%, including very simple kernels like k_{MDED} and k_{MDARC0}. It is also worth to mention that the kernels based on the arc-cosine kernel perform as well if not better than other more common kernel functions. In particular, k_{MDARC1} is the best performing individual kernel, surpassing Global Alignment in the version that uses out-of-sample validation.

By a considerable margin, the best performing method is the combination of kernels created with EasyMKL, surpassing all the individual kernels. This technique tends to over-fit, as it can be observed in the difference of accuracy between training, validation and testing. It is most prominent in the case of cross-validation, were training accuracy is much higher than test and validation accuracy measures. The normalization (scaling individual kernel matrices to have the l2 norm) also has interesting repercussions on the results: it reduces training accuracy and increases validation accuracy, reducing over-fitting. The results of the normalization technique seem to differ depending of the validation technique employed.

Finally, it is also worth to comment the effects of the different validation techniques on the results. There is no clear pattern to determine if cross-validation selects better or worse models than out-of-sample since all kernels react different to this parameter tuning technique. In the cases of k_{GA} and not-normalized MKL the results improve, but in the rest of kernel functions have the same or worst test accuracy. The results clearly differ with [3], specially in the case of k_{VAR}.

Figure 4 shows the resulting weights of the process of EasyMKL. It can be an interesting source of information to visualize how the algorithm determines which kernels are relatively more important.

From the results it can be observed that k_{MDARC1} usually is the kernel with most weight. This further supports the fact that this kernel function is possibly the best performing one for the problem. However, in the case of MKL with out-of-sample validation and normalization, k_{GA} is the kernel with highest weight and this combination is also the one with higher test accuracy. k_{MDARC0} also receives weights higher than the mean, signaling that it is also important in the construction of the model and adds additional information. Finally, it is worth to comment the impact of the normalization in the weights: in all the cases it decreases the weights of k_{VAR} and k_{MDED}, the worst performing kernel functions.

The following table 4 contains the experiments performed with exogenous variables and their results.

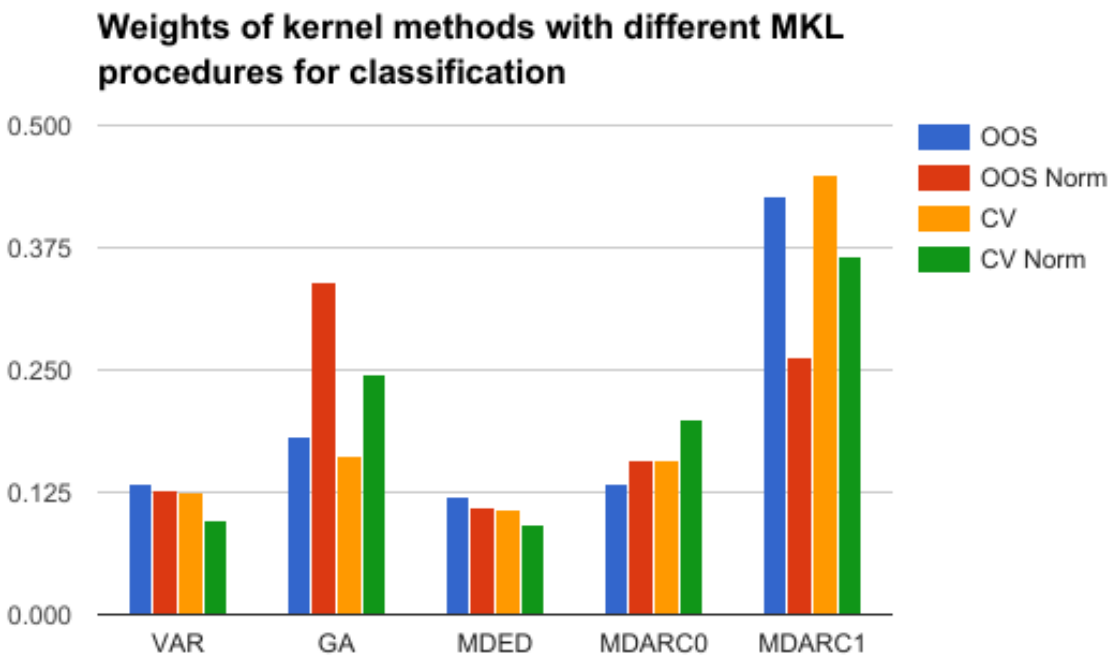

Figure 4: Weights of kernel methods with different MKL procedures for classification

Figure 6.1 reflects the different weights obtained with the MKL process. All the experiments are executed using the best performing kernel method for the data, normalized multiple kernel learning using out-of-sample validation technique. Each variable is contained in a data frame that will be transformed into a kernel matrix using the MKL procedure.

	Train Acc.	Validation Acc.	Test Acc.
EX1	0.757	0.767	0.574
EX2	0.764	0.772	0.567
EX3	0.758	0.784	0.600
EX4	0.733	0.779	0.533

Table 4: Resulting accuracies using exogenous variables.

The best performing method is EX3 in terms of test accuracy. EX1 and EX2 are fairly similar, indicating that the inclusion of the third lag does not affect too much the model. EX4 is the worst performing one, which can mean that the model heavily rely on the lags of SPX to predict the output. The weights of EX1, EX2, and EX3 are near the mean, with slightly higher weight for the endogenous variable. This result does not mean that the rest of variables are not important to predict the result (that will be represented with weights near zero) but that they are mostly equally important. EX4 has its weights shifted towards the exogenous variables. As those data frames lack of lags of SPX it is possible that the information contained in the exogenous variables takes a more active role in the prediction procedure, however the model performs worse in comparison to the rest. It is also worth noting that stock variance is still one of the most relevant

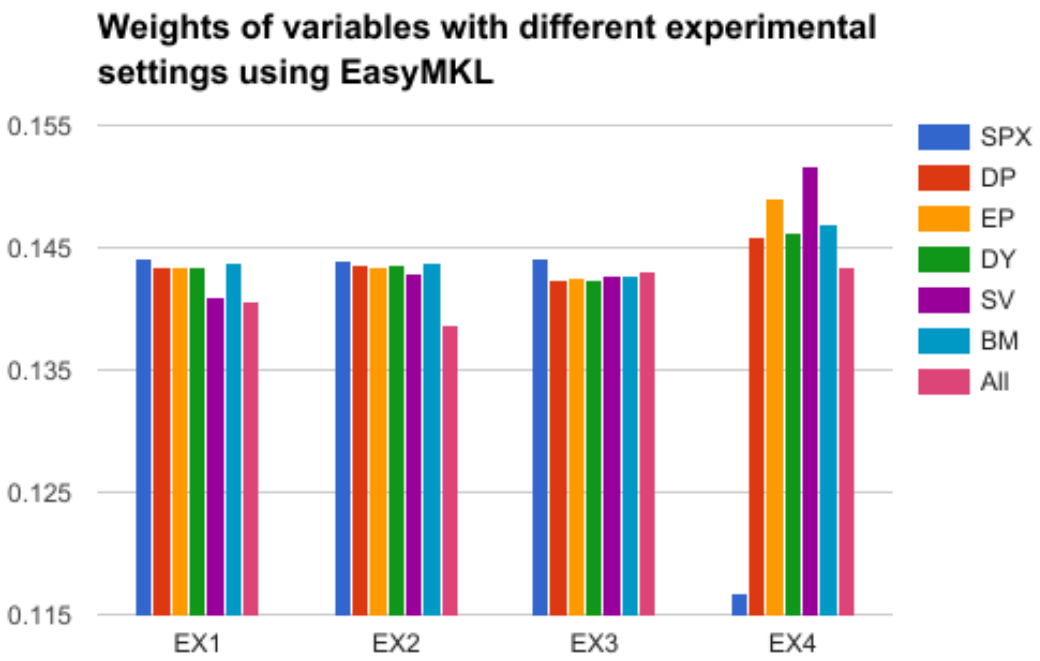

Figure 5: Weights of variables with different experimental settings using EasyMKL

variables in this case.

6.2 Regression task

Tables 5 and 6 shows the evaluation of the different kernel methods and multiple kernel learning using Center Alignment. These results are obtained using the same data for each method but adjusting the parameters individually. In all the methods the impact of using out of sample validation and cross-validation is also tested. All the methods are validated in the same range of parameters. For C and ν the available values are 0.2, 0.4, 0.6, 0.8, and 1. The range of σ is 0.5, 1, 1.5, and 2.

	k_{VAR}	k_{GA}	k_{MDED}	k_{MDARC0}	k_{MDARC1}	MKL	MKL Norm
Train MSE	0.0140	0.0029	0.0039	0.0045	0.0015	0.0032	0.0132
Validation MSE	0.0269	0.0119	0.0171	0.0137	0.0355	0.0091	0.0140
Test MSE	0.0472	0.0295	0.0326	0.0304	0.0645	0.0252	0.0351

Table 5: Results of kernel combinations using Out-Of-Sample Validation.

Mirroring the results of the classification task, the results of regression fall in a determined range that states the capabilities of these methods in the economic prediction tasks. The test errors are in the range between 0,02 and 0,07. In these results higher differences can be observed between the performances of different methods, with some methods being nearly thrice more accurate than others. Simpler methods

	k_{VAR}	k_{GA}	k_{MDED}	k_{MDARC0}	k_{MDARC1}	MKL	MKL Norm
Train MSE	0.0122	0.0014	0.0015	0.0030	0.0009	0.0017	0.0109
Validation MSE	0.0305	0.0141	0.0184	0.0172	0.0326	0.0106	0.0184
Test MSE	0.0434	0.0265	0.0297	0.0275	0.0618	0.0251	0.0313

Table 6: Results of kernel combinations using Cross-Validation.

like k_{MDED} and k_{MDARC0} perform relatively good however k_{MDARC1} has worse results than the rest of the methods, probably attributed to over-fitting looking at the training error.

The best performing technique is the MKL, as it was in the classification case. A clear case of over-fitting can also be observed in this case considering the difference between the training error and the testing error. The parameters have been optimized using a shallow range of values so a fine tuning of them may reduce this effect. The normalization technique only increases the error of the methods, indicating a lose of information during this process.

The different validation techniques have a revealing impact on the regression task. As it can be observed on the tables, the results obtained using cross-validation as a parameter validation technique are always better than the result of OOS. This indicates that this technique is certainly useful for the validation process of the regression task. The results clearly support [3], contradicting the conclusions reached in the classification task.

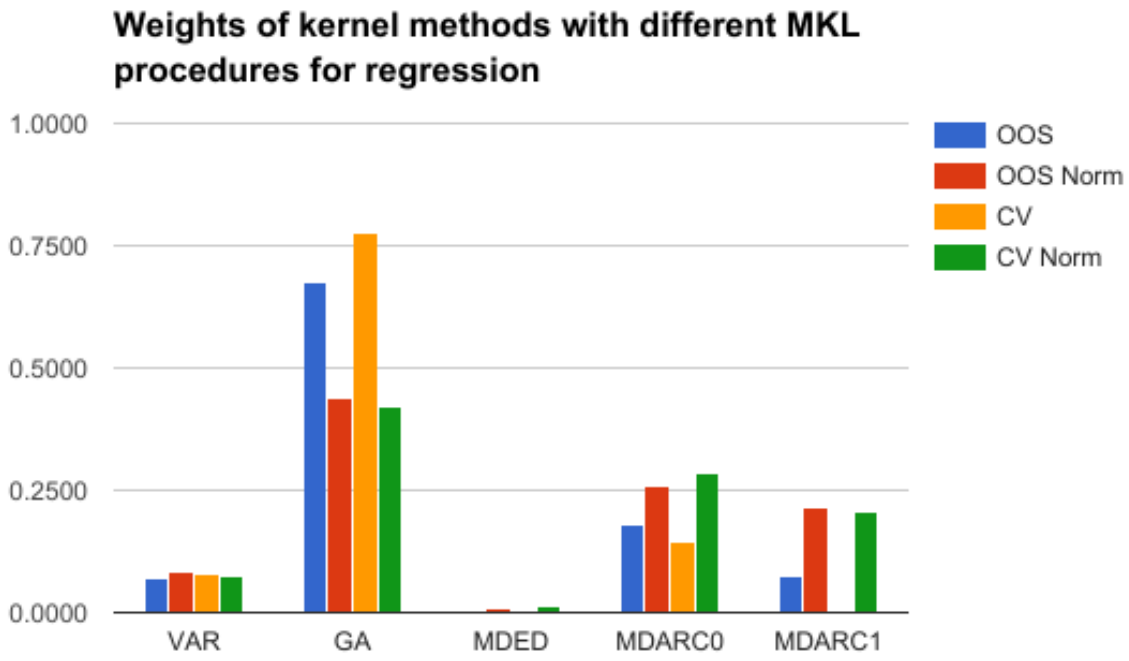

Figure 6: Weights of kernel methods with different MKL procedures for regression.

Figure 6.2 contains the resulting weights of the process of Center Alignment. It can be an interesting

source of information to determine how the algorithm evaluates which kernels are relatively more important. From the results can be observed that k_{GA} obtains the higher percentage of weight in all of the variations of the experiment, with even higher values in the best performing settings of MKL. This fact is reinforced by the better results obtained by this kernel in the individual tests. The normalization procedure impacts the weights by making their values be more similar to the mean and assigning a much higher weight to k_{MDARC1}. These results for k_{MDARC1} are opposite to the ones in the classification task, which indicates how much these tasks are different. k_{MDARC0} has a significant impact on the resulting vector of weights, which further mirrors the individual results.

The table 7 and the figure 7 contain the experiments performed with exogenous variables and their results. All the experiments are executed using the best performing kernel method for the data, non-normalized multiple kernel learning using the cross-validation technique. Each variable is contained in a data frame that will be transformed into a kernel matrix using the MKL procedure. The parameter ranges remain unchanged in these results.

	Train MSE	Validation MSE	Test MSE
EX1	0.0025	0.0129	0.0321
EX2	0.0024	0.0131	0.0332
EX3	0.0024	0.0131	0.0332
EX4	0.0024	0.0135	0.0344

Table 7: Resulting errors using exogenous variables.

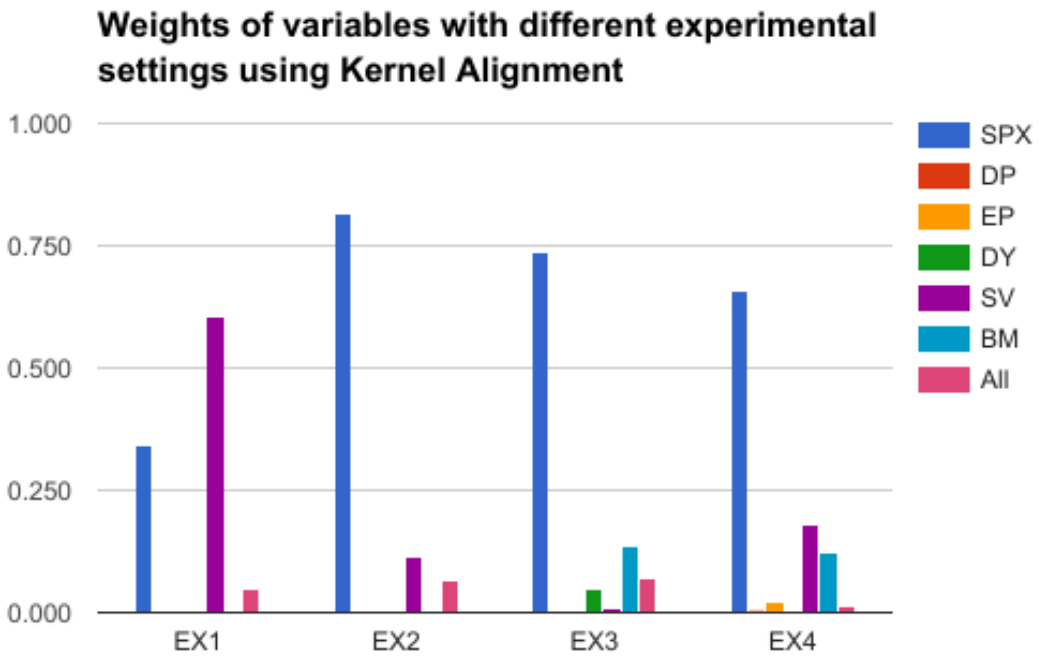

Figure 7: Weights of variables with different experimental settings using Kernel Alignment.

The best performing method in the regression task for these parameters is EX1 in terms of test MSE. There is not much difference between the four experiments in their results, with EX1 being slightly better than the rest. It is noteworthy that all the results are worst than the versions without the exogenous variables. On the other hand, the weights are somewhat different. The SPX index is clearly the feature that most weight has in most of the variations of the problem, ranging between 60% and almost 80%. EX1 is an interesting result, as it has more weight to the Stock Variance than the SPX. This experiment also has the best results in the test set, however those results are bad in comparison to the errors of the endogenous variable. Given those results it seems that the regression task does not give any relevant weight to the exogenous variables in most of the cases, it relies mostly on the SPX feature to carry the prediction task. It is possible that the stock variance has interesting implications on the predictive capabilities, but the results contradict that statement.

Conclusions

The results indicate that, in this experimental procedure, the exogenous variables have a questionable importance in the predictive models. In the case of the classification task the weights are comparable between them, however this means that there is not a clear variable affecting the model more than others. The regression task yields different results, given high weights to the endogenous variable. The overarching conclusion in these results is that exogenous variables do not seem to increase the predictive capabilities of the model. Furthermore, in some cases, they worsen it.

The instability of these results is reported clearly in [40]. Not only the results are worst with the introduction of exogenous variables, but the weights also fluctuate with each different configuration of lags and variables. This can also be attributed to the capabilities of MKL as a learning model. The problem of over-fitting was present in both tasks, reporting low train error measures contrasted by high error scores in the validation and the test sets.

This experimentation concludes that introducing exogenous variables in this procedure does not increase the predictive capabilities of the model. Left to discuss is whether this result is the best that can be obtained from the data or if the reason behind the relatively bad results are the capabilities of the models.

7. Exhaustive Experiment

The comparatively bad results obtained in the previous section discourage the use of exogenous variables however, being one of the objectives of the thesis, it will be interesting to make more tests using those variables. In this section, a new batch of experiments will be explained and executed, with a discussion on its results. In order to ascertain if Multiple Kernel Learning is hindering the results of exogenous variables they will be tested against all the single kernels. The capabilities of the kernels for time series that were showcased in this thesis will also be tested taking advantage of the battery of tests that will be generated against a non-time dependant model in the form of the RBF kernel.

The experimental procedure will be similar to the one introduced in the regression and classification tasks. The data will be used with the same compression, both validation methods will be used, the performance metric will be the same, and the experiments will be performed using the same scrolling window approach. The changes to the procedure are:

- Only regression will be considered: the results from classification may be useful for investment tasks but the classes are created using a threshold, which makes the resulting binary division unstable, hard to predict, and hard to evaluate.
- Normalization will not be used: as seen in the previous results, normalizing the kernels in regression tasks does not yield improvement in the results.
- The data will not be structured in experiments: instead all the data frames created for those experiments will be used individually to test each algorithm.

The main objective of this section is to test if MKL worsens the possible results of using a basic kernels, to determine if the exogenous variables affect the predicting capabilities of the model in a positive way, and to ascertain if it is possible to apply the Cross-Validation technique as a parameter optimization procedure. In order to decide this facts, a factorial experiment will be performed.

7.1 Radial Basis Function Kernel

One of the most used kernel functions in the practice is the Radial Basis Function[39]. It is employed for general data inputs and as a default kernel in most of the Support Vector Machine implementations. It uses the squared Euclidean distance to calculate the dissimilarity between samples. Generally outputs good results with diverse types of data. The most common formulation is the following:

$$K(x, x') = exp\left(-\frac{\|x - x'\|^2}{2\sigma^2}\right), \tag{34}$$

where σ is a free parameter. RBF, however, does not work directly with time series as it needs data of the same size in order to compare it. The data set of Welch is almost complete, as its data is measured monthly. Some entries are missing in the beginning (some variables were not recorded as early as the SPX), but they can be discarded in order to have a complete data set that can be feed to a RBF kernel.

7.2 Factorial Design

The Factorial Design[5] is a methodology to define tests and to compare results. In problems that have several decisions to be made and several possible combinations of them this methodology provides the design principles to build a group of tests to reach a hypothesis about the results. This procedure determines if a decision is statistically better than other and if there are relevant interactions between decisions.

In the particular case of this thesis there could be a high number of decisions suggested in the previous section: normalization of the kernels, use of different lags of each variable, the parameters of the kernels, the parameters of the kernel learning algorithms, etc. Those topics, however, will be left out of the scope of this document as factorial experiments increases exponentially the number of tests to do when new decisions are introduced. The final decisions of the factorial experiments will be the following:

- Use MKL or any single kernel methods
- Use any combination of exogenous variables or only the endogenous variable
- Use Cross-Validation or not

Even though the decisions are clear, the number of factors on each decision is not so straight forward. The setting has several single kernels and several exogenous variables. Which of them should be used and how should they be used is a problem by itself. In order to have insightful but time-conscious results they must be considered in the methodology with several restrictions.

The data is divided in all the data frames created for the previous experiments. This creates a significant amount of data to explore and test each of the methods. The different sets of data are defined in 8. The inclusion of some data sets may seem unreasonable as they share similar structure, however they were all defined for various tests in the code and their impact could be seen as enriching information, hence their inclusion.

7.3 Experimental Results

Before using the formulas of the factorial experiment to analyze the decisions and the results it is imperative to set the results that will be considered in comparison. The experimental procedure employed uses several different representations of the same data with several different methods. In order to use the factorial design, in the most clear and simple version, it is necessary to narrow the results to one by decision. In this section the results will be commented and a model will be selected for the factorial experiment comparison.

The results obtained can be found in Appendix B, which are plenty and cumbersome to interpret. The parameter ranges defined in the regression task section remain unchanged in these results. In the lines of the decisions presented in the experimental definition, the following a priori conclusions can be obtained:

- The best result of the MKL method is surpassed by the best results of the RBF and MDARC0 kernels by a very slim margin. (0.02505 vs 0.02448 and 0.02497)
- All kernel methods except MKL perform better with the inclusion of exogenous variables, but with different degree of improvement.
- Cross-Validation seems to make general but small improvements applied with any kernel function.

	SPX	DP	EP	DY	SVAR	BM
0	0:4					
1	0:4	0				
2	0:4		0			
3	0:4			0		
4	0:4				0	
5	0:4	0	0	0	0	
6	0:4	0, 3				
7	0:4		0, 3			
8	0:4			0, 3		
9	0:4				0, 3	
10	0:4	0, 3	0, 3	0, 3	0, 3	
11	0:4	0:4				
12	0:4		0:4			
13	0:4			0:4		
14	0:4				0:4	
15	0:4	0:4	0:4	0:4	0:4	
16	0	0:4				
17	0		0:4			
18	0			0:4		
19	0				0:4	
20	0	0:4	0:4	0:4	0:4	
21	0:4					0:4
22	0					0:4
23	0:4					0
24	0:4	0	0	0	0	0
25	0:4					0, 3
26	0:4	0, 3	0, 3	0, 3	0, 3	0, 3
27	0:4	0:4	0:4	0:4	0:4	0:4
28	0	0:4	0:4	0:4	0:4	0:4

Table 8: The definition of the data frames to test the variables.

In terms of the decision of using MKL or not the results of the RBF kernel are surprising, as it surpasses the much more complex MKL algorithm. MDARC0 also gets an improvement by using the exogenous variables and it becomes the single best performing, time-dependent kernel function. The main difference between MDARC0 and MDARC1 relates again to the over-fitting, as can be observed in the minuscule training error of MDARC1 for its best result. The ratio between the test error and the train error can be a useful tool to determine the degree of over-fitting. This ratio will be defined simply as $RatioofOverfitting = TestError/TrainingError$ and quantifies how much the model fits to the training data. The results of this comparison can be found at figure 8. As the figure shows the RBF and MDARC0 kernels suffer less than the half of over-fitting than MKL. It is possible that this fact means that the MKL model produces worst results as it falls on this problem. The notable exception is VAR which has the lowest ratio, however it outputs noticeably worst results, perhaps a result of bad fitting.

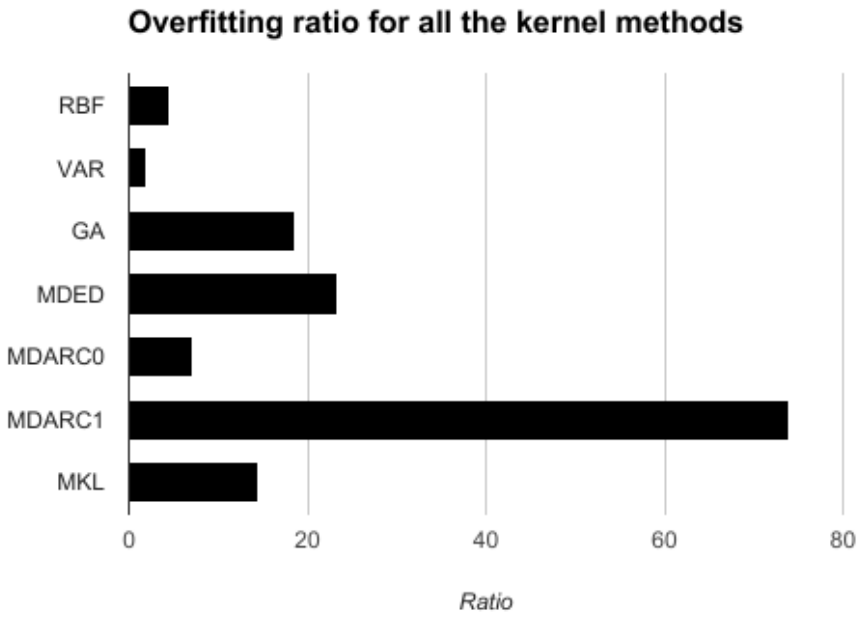

Figure 8: Ratio of over-fitting for all kernel methods

As shown in the regression task, the inclusion of exogenous variables in the MKL model decreases the predictive capacity. However the results reported in Appendix B show that every other kernel method increases their predictive capacity by including this additional data. The most notable case, again, is the fact that RBF improves the most compared with other methods shifting its results from 0.0293 using only endogenous variables to 0.0245. Other improvements near the 15% performance increase can be appreciated in the VAR kernel. MDARC0 also improves in about 10% its test MSE. The rest of kernel methods only increase their results by 5% or less. Another sources of interesting information are the data sets with which the models obtain said increases in performance. The data sets are 1, 2, 17, 18, and 22. In the case of 1 and 2 they are selected by TGA and MDARC1, the models that tend to over-fitting and not improve much with the exogenous variables. The rest of data sets have common facts, which are the lack of lags of the SPX index (i.e. the endogenous variable) and that all of them include a single exogenous variable with four lags (Dividend Price, Dividend Yield and Book-to-market ratio). This contrasts with

Factor Levels	-	+
1 Kernel Method(KM)	RBF	MKL
2 Variables(Var)	Endogenous	Exogenous
3 Validation Method(VM)	OOS	CV

Table 9: The different factor levels for each decision of the Factorial Experiment.

the clear tendency of MKL to favor the SPX index with maximum lags. It is safe to conclude that, in the methods with increased performance by the introduction of exogenous variables, the endogenous variable has a reduced impact contrary to the methods that saw little or no impact in the use of exogenous variables. Looking at the results it is possible that the models like MKL and MDARC1 over-fit towards the SPX index as it is the most similar to the output variable, but the models like RBF and MDARC0 unravel correlations between the output and the exogenous variables that the other methods apparently do not do.

The decision between using or not the Cross-Validation technique is also looked upon. As introduced in [3] Cross-Validation is not demonstrated to work with non-independent signals, non-stationary distributions of data and/or non-auto-regressive models. The results of this experimental approach indicate positive results using this technique. All of the kernel methods get increases in performance when this technique is employed, and most of them also had their best results using it. The only exception is RBF which, in general, improves with CV however the best result is found using OOS by a slim margin. Although the improvement of this technique is empirically true, the results are not conclusive on the effects that this technique can have in similar data sets or methodologies. The effects are slim however, less than a 10% improvement in all the cases.

7.4 Factorial Experiment Results

Although testing all the possible combinations of models, exogenous variables, and validation techniques is extremely tempting to create conclusive comparisons, the sheer number of tests scales quickly with the introduction of so many factors. Considering the seven different methods, the six exogenous variables that can be combined between them, and the two validation techniques the number of test grows to 896 ($7 * 2^6 * 2$). This bulk of computational expense is unfeasible on the scope of this work so, in order to reach conclusive results, a simplification should be done.

The decisions are reduced to two levels(options) per each. The first decision will be reduced to use MKL or the best performing single kernel method, which is RBF. The second decision will only consider the best performing exogenous variable combination for each method. The third decisions will be unaltered, testing both validation techniques. Following the representation procedures of the factorial experiment, the different levels of each decision are represented as in table 9 and the different formulations (experiments) are represented in table 10 along with the test MSE of each experimental combination with the variance of the results.

Next on the factorial experiment procedure are the equations that determine the influence of each decision and the interaction of those decisions. The formulation is quite simple for the individual comparisons between levels: the mean of one level across all the experiments minus the mean of the other level across all the experiments. The formulation of the interaction between two variables is more complex: the mean of the experiments where both decisions have the same level symbol (both + or -) minus the mean of the experiments where both decisions have different level symbols (if one is + the other must be - and viceversa). The interaction between three levels is even more intricate: obtain the difference between each

Formulation	1	2	3	Result	Variance
1	-	-	-	0.02930723	0.01210860
2	+	-	-	0.02519840	0.01040213
3	-	+	-	0.02448274	0.00897466
4	+	+	-	0.02918150	0.01196361
5	-	-	+	0.02840321	0.01228031
6	+	-	+	0.02505082	0.01156009
7	-	+	+	0.02500117	0.00982782
8	+	+	+	0.02729349	0.01146790

Table 10: The results of each experiment of the Factorial Experiment.

level of decision 1 fixing the rest of decisions($y_8 - y_7$, $y_6 - y_5$, etc), subtract those results using the levels of the second variable fixing the third decision ($(y_8 - y_7) - (y_6 - y_5)$, etc), and finally subtract the mean of each level of the third decision. The results of these equations for the current problem are the following:

$$KM = \frac{0.02519 + 0.02505 + 0.02918 + 0.02729}{4} - \frac{0.0293 + 0.02840 + 0.02448 + 0.025}{4} = -0.000117$$
$$Var = \frac{0.02448 + 0.025 + 0.02918 + 0.02729}{4} - \frac{0.02931 + 0.0284 + 0.0252 + 0.02505}{4} = -0.000500$$
$$VM = \frac{0.02931 + 0.02448 + 0.0252 + 0.02918}{4} - \frac{0.02840 + 0.025 + 0.02505 + 0.02729}{4} = 0.000605$$

$$KM\&Var = \frac{0.02931 + 0.0284 + 0.02918 + 0.02729}{4} - \frac{0.02448 + 0.025 + 0.0252 + 0.02505}{4} = 0.003613$$
$$KM\&VM = \frac{0.02931 + 0.02448 + 0.02505 + 0.02729}{4} - \frac{0.0284 + 0.025 + 0.0252 + 0.02918}{4} = -0.000412$$
$$Var\&VM = \frac{0.02931 + 0.0252 + 0.025 + 0.02729}{4} - \frac{0.02448 + 0.02918 + 0.02841 + 0.02505}{4} = -0.000079$$

$$KM\&Var\&VM = \frac{\frac{(0.02729-0.025)-(0.02505-0.0284)}{2} - \frac{(0.02918-0.02448)-(0.0252-0.02931)}{2}}{2} = -0.000790$$

The variance is also considered on the formulations of the factorial experimentation. It generates intervals of confidence of the results, which should be taken into account to ascertain if the decision being analyzed has a substantial impact on the results. As the number of variance measures obtained is 8, that means that the measurement will have 8 degrees of freedom. Given the results and its variance it is possible to determine if the observed effects of the decisions can be attributed only to noise. The authors of the experimental procedure indicate a change of 2 to 3 times the variance in the decisions to be considered as a potential change in performance, but ultimately suggest that these intervals should be set by the reader looking at the t-distribution table. Looking strictly at the obtained variance, the mean variance of all the formulations is 0.01107314, the variance of each effect is $V(effect) = (\frac{1}{8} + \frac{1}{8}) * 0.01107314 = 0.002768285$ and the standard error of each effect is $SE(effect) = \sqrt{V(effect)} = 0.0526144942$.

The table of calculated effects is 11. The effects of these decisions are very small, even including the interactions between them. This indicates that the decisions taken do not change the capabilities of the model by any significant stretch, which is an expected result looking at the very small differences between

Name	Effect with standard error
Kernel Methods(KM)	$-0.000117 \pm 0.0526144942$
Variables(Var)	$-0.000500 \pm 0.0526144942$
Validation Method(VM)	$0.000605 \pm 0.0526144942$
KM x Var	$0.003613 \pm 0.0526144942$
KM x VM	$-0.000412 \pm 0.0526144942$
Var x VM	$-0.0000797 \pm 0.0526144942$
KM x Var x VM	$-0.000790 \pm 0.0526144942$

Table 11: Table of calculated effects

the compared methods. Without looking at the standard errors it can be seen that: MKL yields worst results than RBF on average, the same can be said for the use of exogenous variables over endogenous variables, and the cross-validation technique is better on average than out-of-sample validation. These results, however, pale in comparison with the massive variance of the model. None of the observed results are even capable of coming close to this metric, indicating that the influence of the noise is quite big. Even though the results may be highly influenced by noise there is an interesting result. The interaction between the kernel methods and the selected economic variables is notable and this effect can appear in other versions of the problem, and be further explored.

Exhaustive Analysis Conclusions

The results of the factorial experiment indicate that most of the possible improvements that can be observed in the MSE of these decisions are ultimately not reliable due to the high amount of noise in the samples. This unreliability in the models was indicated previously in [40] and again demonstrated in these experiments. Some interesting results, however, must be addressed: MKL integrating five kernels for time-series is matched by a simple RBF and also by one of its composing kernels, MDARC0. Furthermore, those simple models achieve the predictive capabilities of MKL using exogenous variables. The interaction between the kernel methods and the exogenous variables can be also observed in the relatively high interaction between those decisions. Cross-validation also raises as a possible better validation algorithm than Out-Of-Sample. This does not contradict the theorem proposed in [3], but these empirical observations lack of theoretic proof and can be ultimately spurious given the influence of noise.

The performance of MKL is highly put in question during these experiments, it does not improve while presented with exogenous variables and over-fits to the training data. It is possible to think that the selected algorithm does not comply with the data or the definition of the problem however, if the weights produced by the algorithm are ignored and proportional weights are used, the predictive capabilities of the model are highly reduced(from a MSE of 0.02505082 to 0.053655275643).

During all of the experimental procedure the developed algorithms were not compared with a ground truth. In [40] they use a linear model to create the predictors for their data. Using a basic implementation it is possible to apply a very similar linear model to these data. The results of this experiment are reported in B.8. Some basic observations can be done by looking at the table: the linear model reports several times worse results than the studied algorithms, and the exogenous variables introduce positive changes in the predictive capabilities of the model. The comparison of this results to the ones obtained from the best performing model of the present work can be found at figure 9. The best result of the kernel methods is RBF with a test MSE of 0.02448274 against the best result of the linear model with a MSE of 0.09958952476. Although the difference of performance may very possible be affected by the noise of the

data it is also significant enough to be a real improvement.

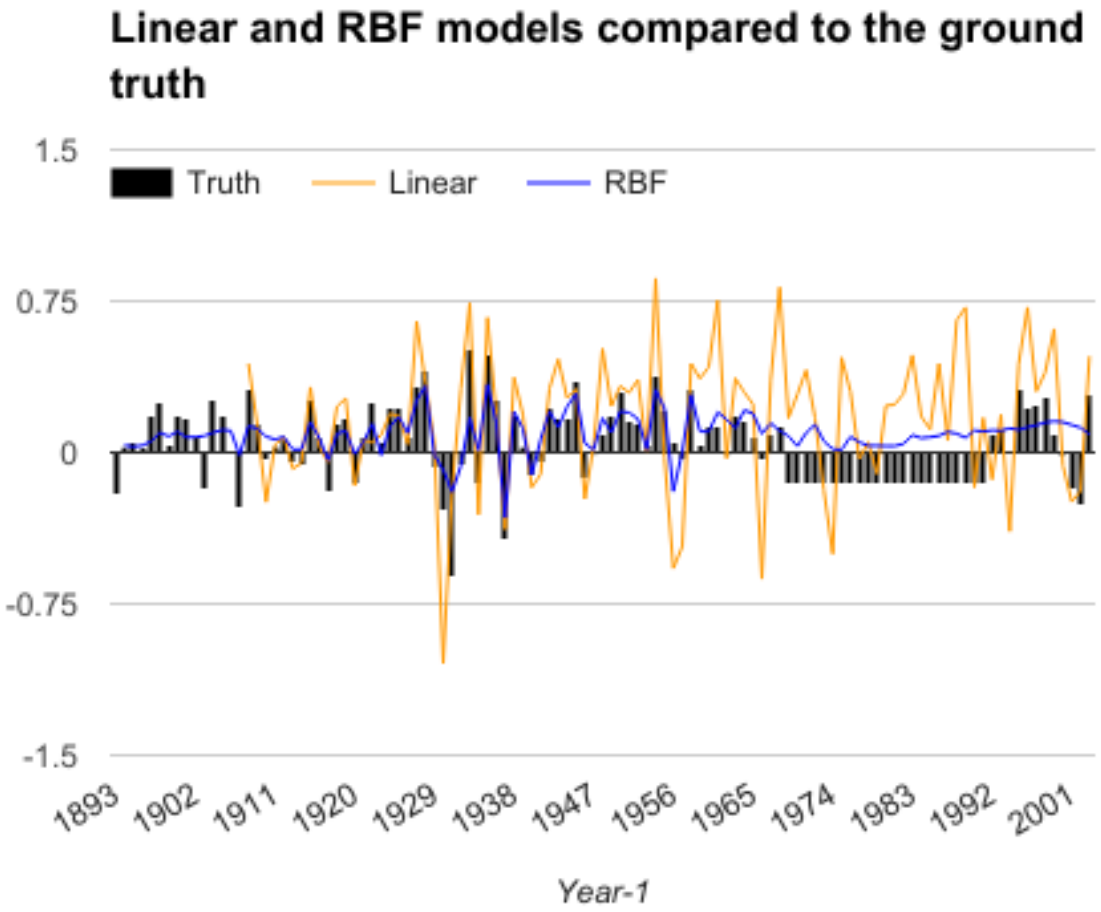

Figure 9: Comparison of both methods against the ground truth.

8. Conclusions

In this thesis, a Multiple Kernel Learning approach is applied to the Goyal and Welch experimental data [40] in order to determine the capabilities of this technique over financial data, in particular on the prediction of the SPX Equity Premium. Alongside it is also studied if a set of financial variables (called exogenous variables in this thesis) and if the Cross-Validation technique (as part of the parameter tuning process) improve or not the quality of the models. The experiment is designed around several kernel functions for time series that aimed to extract relevant information from these variables and create a predictive model. The experimental procedure is based on selecting the best kernel or combination of them for this problem, applying the selected model to each of the variables creating several kernel matrices, and obtaining the multiple kernel learning weights of each of those matrices. Those weights theoretically indicate the relative importance of each variable.

Multiple Kernel Learning results to be an algorithm that obtains interesting results when the data is similar to the output, reporting better predictions than its composing kernels. However, when exogenous variables are introduced, the MKL technique is not capable of exploiting the composing kernel methods that extract the best from those variables. This algorithm also suffers from over-fitting and long computational times to create each kernel matrix and integrate them. The algorithms present in this thesis clearly work, as the test performed with the mean weight yields worse MSE. MKL seems not to be a reliable model for this type of data and it does not use the capabilities of the underlying kernels. As future work it is possible to explore the possible kernel matrices that compose the MKL, change the MKL algorithm, or its parameters to obtain better results.

The impact of the exogenous variables is unclear. While it has notable effects on several models the noise of the data(normally present on financial data sets) makes it difficult to determine if they improve or not the models. If the noise of the data is assumed, the best performing kernel methods use these variables. In particular the Dividend Yield and the Earning to Price with no lags of the SPX help to obtain the best results. It is also observed that the better performing models use zero lags of the SPX whilst the models that suffer from over-fitting do the contrary. The use of exogenous variables shows the possibility of a factual improvement, but the nature of the data and the previous research suggest the contrary. As future work the rest of the variables presented on [40] can be considered, different configurations of lags and variables can be tested, and different models can be adopted.

Cross-Validation also falls on the same problems as the rest of the decisions, the noise of the data makes it difficult to make solid statements about its capacities on this problem. Ignoring the noise, CV seems to improve the predictive capabilities of the models even though the data is surely non-stationary. This does not contradict the hypothesis formulated in [3] but these results do not have a theoretical base. CV should be further explored on time series as it provides a better use of the data and redundancy on the parameter validation. Future work on this topic will include looking into different types of data, and measuring the capabilities of this validation method.

The Radial-Basis Function Kernel is the surprise of these experiments. Not only it is capable to unravel the potential use of the exogenous variables but it is also able to equal MKL and other time dependent methods. It is an interesting case, as the predictive capabilities of the model are low with endogenous variables and high with exogenous. It is also much faster computationally than other kernel methods and very easy to apply. The results however are also dependent on the noise of the data and are not conclusive, but the effects of the additional data are considerable. Future work on RBF for time series include looking into other data sets, and creating other alignments.

The applied methods surpass easily linear models which indicate that, regardless the noise, the models clearly work and offer a better predictive capability. All of the presented methods can be improved and further tuned to increase their usefulness even more, creating a bigger gap with simple solutions and showing the potential of machine learning methods on the prediction of the market returns.

References

[1] Aiolli, F., Donini, M.: EasyMKL: a scalable multiple kernel learning algorithm. Neuro computing 169, 215224. (2015)

[2] Ang, A., and Bekaert, G. (2007). Stock return predictability: Is it there?. Review of Financial studies, 20(3), 651–707.

[3] Bergmeir, C., Hyndman, R., Koo, B.: A Note on the Validity of Cross-Validation for Evaluating Time Series Prediction. Department of Econometrics and Business Statistics, Working Paper, ISSN 1440-771X (2015)

[4] Bhattacharya, J. and Kanjilal, P. P.: On the Detection of Determinism in a Time Series. Phys. D, vol.132, no.1-2, p.100-110, doi.10.1016/S0167-2789(99)00033-0. (1999)

[5] Box, George E. P., Hunter, J. Stuart, and Hunter, William G.: Statistics for Experimenters: Design, Innovation, and Discovery, 2nd Edition ISBN: 978-0-471-71813-0, 664 pages. (2005)

[6] Campbell, J. Y., and Thompson, S. B. (2008). Predicting excess stock returns out of sample: Can anything beat the historical average?. Review of Financial Studies, 21(4), 1509-1531.

[7] Campbell, John Y., and Shiller, R. J. (1988). The dividend-price ratio and expectations of future dividends and discount factors, Review of Financial Studies 1, 195–228.

[8] Chang, C., Lin, C.: Training ν-Support Vector Classifiers: Theory and Algorithms. Neural Comput., vol. 13, no. 9, 2119–2147, ISSN 0899-7667, September (2001)

[9] Chang, C., Lin, C.: Training ν-support Vector Regression: Theory and Algorithms. Neural Comput., vol. 14, no. 8, 1959–1977, ISSN 0899-7667, August (2002)

[10] Cho, Y., Saul, L.: Kernel Methods for Deep Learning. Advances in Neural Information Processing Systems 22, 342–350, Curran Associates, Inc. (2009)

[11] Cochrane, John H. (1992). Explaining the variance of price-dividend ratios, Review of Financial Studies 5, 243–280

[12] Cochrane, John H. (2006). The dog that did not bark: A defense of return predictability, Review of Financial Studies 21, 1533–1575.

[13] Cochrane, John H. (2011). Presidential Address: Discount Rates, The Journal of Finance 56 (4), 1047–1108.

[14] Cortes, Corinna., Mohri, Mehryar. and Rostamizadeh, Afshin.: Algorithms for Learning Kernels Based on Centered Alignment Journal of Machine Learning Research 13, 795-828. (2012)

[15] Cuturi, M.: Fast Global Alignment Kernels. Graduate School of Informatics, Kyoto University. (2011)

[16] Dickey, David A., and Wayne A. Fuller. "Distribution of the Estimators for Autoregressive Time Series With a Unit Root." Journal of the American Statistical Association 74, no. 366 (1979): 427-31. doi:10.2307/2286348.

[17] Duan, W.-Q. and Stanley, H. E. (2011). Cross-correlation and the predictability of financial return series. Physica A: Statistical Mechanics and its Applications, 390(2):290 296.

[18] Fama, E. F. (1970). Efficient capital markets: A review of theory and empirical work*. The journal of Finance, 25(2):383417.

[19] Fama, Eugene F., and French, Kenneth R. (1988). Dividend yields and expected stock returns, Journal of Financial Economics 22, 3–25.

[20] Favero, Carlo A.: Applied Macroeconometrics. New York: Oxford University Press, pp. 162213. (2001)

[21] Fletcher, T. (2012). Machine learning for financial market prediction. PhD thesis, UCL (University College London).

[22] Fung, D. S.: Methods for the Estimation of Missing Values in Time Series. Master Thesis, School Of Mathematics and Engineering, Edith Cowan University. (2006) http://ro.ecu.edu.au/theses/63

[23] Gnen, Mehmet. and Alpaydn, Ethem.: Multiple Kernel Learning Algorithms. Jour. Mach. Learn. Res. 12(Jul):22112268, (2011)

[24] Hansen, Lars Peter, and Hodrick, Robert J. (1980). Forward exchange rates as optimal predictors of future spot rates: An econometric analysis, Journal of Political Economy 88, 829– 853.

[25] Hofmann, Thomas; Scholkopf, Bernhard; Smola, Alexander J.: Kernel Methods in Machine Learning. The Annals of Statistics, Vol. 36, No. 3, 11711220. (2008)

[26] Hyndman, Rob J.: Detecting seasonality. http://robjhyndman.com/hyndsight/detecting-seasonality/ (Accessed in 28/11/2016)

[27] Kim, K.-j. (2003). Financial time series forecasting using support vector machines. Neurocomputing, 55(1):307319.

[28] Kothari, S. P., and Shanken, J. (1997). Book-to-market, dividend yield, and expected market returns: A time-series analysis. Journal of Financial Economics, 44(2), 169-203.

[29] Lettau, M., and Ludvigson, S. (2001). Consumption, aggregate wealth, and expected stock returns. the Journal of Finance, 56(3), 815–849.

[30] Muller, K. R., Smola, A. J., Ratsch, G., Scholkopf, B., Kohlmorgen, J., and Vapnik, V. (1999). Using support vector machines for time series prediction. Advances in kernel methodssupport vector learning, MIT Press, Cambridge, MA, pages 243254.

[31] Nason., G.P.: Statistics in Volcanology Chp.11. University of Bristol. (2006) http://www.cas.usf.edu/ cconnor/geolsoc/html/chapter11.pdf

[32] Peña, M., Arratia, A., Belanche, Ll.: Multivariate Dynamic Kernels for Financial Time Series Forecasting. Dept. of Computer Science, Technical University of Catalonia. (2016) http://upcommons.upc.edu/handle/2099.1/26418?locale-attribute=en

[33] Sakoe, H. and Chiba, S.: A similarity evaluation of speech patterns by dynamic programming. Nat. Meeting of Institute of Electronic Communications Engineers of Japan, p. 136. (1970)

[34] Shiller, Robert J. (1981). Do stock prices move too much to be justified by subsequent changes in dividends? American Economic Review 71, 421–436.

[35] Smola, Alex J. and Schlkopf, Bernhard.: A Tutorial on Support Vector Regression. Statistics and Computing (2003).

[36] Tay, F. E. and Cao, L. (2001). Application of support vector machines in financial time series forecasting. Omega, 29(4):309317.

[37] Tsiporkova, Elena.: Dynamic Time Warping Algorithm for Gene Expression Time Series. www.psb.ugent.be/cbd/papers/gentxwarper/DTWAlgorithm.ppt, Accessed: 2016-12-11.

[38] Vapnik, Vladimir N. and Kotz, Samuel.: Estimation of Dependences Based on Empirical Data. Springer, ISBN 0-387-30865-2, 510 pages. (2006)

[39] Vert, Jean-Philippe.; Tsuda, Koji.; and Schlkopf, Bernhard.; A primer on kernel methods. Kernel Methods in Computational Biology. (2004)

[40] Welch, I., Goyal, A.: A Comprehensive Look at The Empirical Performance of Equity Premium Prediction. National Bureau of Economic Research, 10483 (2004)

A. Appendix: Classification Task Parameters

Method	Nu Range	Sigma Range	Lamb Range
VAR OOS	From 5e-5 to 5e-4 with 5e-5 steps	0.5, 1, 1.5, 2	
VAR CV	From 5e-5 to 5e-4 with 5e-5 steps	0.5, 1, 1.5, 2	
GA OOS	From 5e-5 to 5e-4 with 5e-5 steps	0.5, 1, 1.5, 2	
GA CV	From 5e-5 to 5e-4 with 5e-5 steps	0.5, 1, 1.5, 2	
MDED OOS	From 5e-5 to 5e-4 with 5e-5 steps	0.5, 1, 1.5, 2	
MDED CV	From 5e-5 to 5e-4 with 5e-5 steps	0.5, 1, 1.5, 2	
MDARC0 OOS	From 5e-5 to 5e-4 with 5e-5 steps	0.5, 1, 1.5, 2	
MDARC0 CV	From 5e-5 to 5e-4 with 5e-5 steps	0.5, 1, 1.5, 2	
MDARC1 OOS	From 5e-7 to 5e-6 with 5e-7 steps	0.5, 1, 1.5, 2	
MDARC1 CV	From 5e-7 to 5e-6 with 5e-7 steps	0.5, 1, 1.5, 2	
EasyMKL OOS	From 5e-4 to 5e-3 with 5e-4 steps	0.5, 1, 1.5, 2	From 0.1 to 1 with 0.2 steps
EasyMKL OOS Norm	From 5e-5 to 5e-4 with 5e-5 steps	0.5, 1, 1.5, 2	From 0.1 to 1 with 0.2 steps
EasyMKL CV	From 5e-5 to 5e-4 with 5e-5 steps	0.5, 1, 1.5, 2	From 0.1 to 1 with 0.2 steps
EasyMKL CV Norm	From 5e-3 to 5e-2 with 5e-3 steps	0.5, 1, 1.5, 2	From 0.1 to 1 with 0.2 steps

Experiment	Nu Range	Sigma Range	Lamb Range
EX1	From 5e-3 to 5e-2 with 5e-3 steps	0.5, 1, 1.5, 2	From 0.1 to 1 with 0.2 steps
EX2	From 5e-3 to 5e-2 with 5e-3 steps	0.5, 1, 1.5, 2	From 0.1 to 1 with 0.2 steps
EX3	From 5e-3 to 5e-2 with 5e-3 steps	0.5, 1, 1.5, 2	From 0.1 to 1 with 0.2 steps
EX4	From 5e-3 to 5e-2 with 5e-3 steps	0.5, 1, 1.5, 2	From 0.1 to 1 with 0.2 steps

B. Appendix: Exhaustive Experiment Results

B.1 RBF Model

Exp. No.	Val. Method	Train MSE	Test MSE	Val. Method	Train MSE	Test MSE
0	OOS	0.00306411	0.02930723	CV	0.00103674	0.02840321
1	OOS	0.00349565	0.02835293	CV	0.00123333	0.02746420
2	OOS	0.00350171	0.02837131	CV	0.00123225	0.02745769
3	OOS	0.00349352	0.02836194	CV	0.00123535	0.02747099
4	OOS	0.00286571	0.03029962	CV	0.00059418	0.03019455
5	OOS	0.00266325	0.02869388	CV	0.00096415	0.02855284
6	OOS	0.00352995	0.02780917	CV	0.00142519	0.02691156
7	OOS	0.00358537	0.02784800	CV	0.00143158	0.02692488
8	OOS	0.00353107	0.02781486	CV	0.00142337	0.02691171
9	OOS	0.00283623	0.02956674	CV	0.00069497	0.02938819
10	OOS	0.00338679	0.02838164	CV	0.00163665	0.02719797
11	OOS	0.00385586	0.02705715	CV	0.00203146	0.02603252
12	OOS	0.00391688	0.02732228	CV	0.00205417	0.02599775
13	OOS	0.00384719	0.02698429	CV	0.00203669	0.02595329
14	OOS	0.00291104	0.02853820	CV	0.00116093	0.02803924
15	OOS	0.00509759	0.02968128	CV	0.00332226	0.02773096
16	OOS	0.00551052	0.02448713	CV	0.00402825	0.02498475
17	OOS	0.00570985	0.02516123	CV	0.00415020	0.02559651
18	OOS	0.00549755	0.02448274	CV	0.00401582	0.02500117
19	OOS	0.00474816	0.02594415	CV	0.00323149	0.02640658
20	OOS	0.01005784	0.03217500	CV	0.00851560	0.03151288
21	OOS	0.00254913	0.03107964	CV	0.00059067	0.03084448
22	OOS	0.00342827	0.03058733	CV	0.00161112	0.03100310
23	OOS	0.00223595	0.03166996	CV	0.00024696	0.03168141
24	OOS	0.00254007	0.03106635	CV	0.00058541	0.03084102
25	OOS	0.00233346	0.03138113	CV	0.00029305	0.03141906
26	OOS	0.00303194	0.03134441	CV	0.00096389	0.03034614
27	OOS	0.00448832	0.03125163	CV	0.00180096	0.03076520
28	OOS	0.00731371	0.03450099	CV	0.00560320	0.03377273

B.2 VAR Model

Exp. No.	Val. Method	Train MSE	Test MSE	Val. Method	Train MSE	Test MSE
0	OOS	0.01403406	0.04693318	CV	0.01227778	0.04330353
1	OOS	0.01408445	0.04694520	CV	0.01219379	0.04324724
2	OOS	0.01404446	0.04684107	CV	0.01226774	0.04328547
3	OOS	0.01402557	0.04695789	CV	0.01225121	0.04326257
4	OOS	0.01545896	0.05033875	CV	0.01391573	0.04633112
5	OOS	0.01552038	0.05024569	CV	0.01407337	0.04637820
6	OOS	0.01401942	0.04696514	CV	0.01220506	0.04328864
7	OOS	0.01405244	0.04677887	CV	0.01226669	0.04334671
8	OOS	0.01401965	0.04695247	CV	0.01227670	0.04326884
9	OOS	0.01543887	0.05033083	CV	0.01393131	0.04633275
10	OOS	0.01545542	0.05017743	CV	0.01426716	0.04548493
11	OOS	0.01406075	0.04683115	CV	0.01220627	0.04331017
12	OOS	0.01373267	0.04658093	CV	0.01235272	0.04242848
13	OOS	0.01403474	0.04693739	CV	0.01227131	0.04330260
14	OOS	0.01552793	0.05027838	CV	0.01392910	0.04633126
15	OOS	0.01500905	0.04995592	CV	0.01390806	0.04539797
16	OOS	0.01798986	0.04839287	CV	0.02121137	0.04351260
17	OOS	0.01780996	0.04938047	CV	0.02103073	0.04433980
18	OOS	0.01767696	0.04823614	CV	0.02091500	0.04362724
19	OOS	0.01620554	0.04530918	CV	0.01403630	0.04314954
20	OOS	0.01841130	0.05122172	CV	0.02202310	0.04657330
21	OOS	0.01352386	0.04247674	CV	0.01085813	0.04040997
22	OOS	0.01603756	0.04139779	CV	0.01862742	0.03746968
23	OOS	0.01373443	0.04248061	CV	0.01087158	0.04043157
24	OOS	0.01352945	0.04257538	CV	0.01086651	0.04044086
25	OOS	0.01352907	0.04254827	CV	0.01086894	0.04043085
26	OOS	0.01352562	0.04254171	CV	0.01086153	0.04042571
27	OOS	0.01350713	0.04243564	CV	0.01064099	0.04038666
28	OOS	0.01527347	0.04445001	CV	0.01828113	0.04002803

B.3 GA Model

Exp. No.	Val. Method	Train MSE	Test MSE	Val. Method	Train MSE	Test MSE
0	OOS	0.00291810	0.02938741	CV	0.00141682	0.02656155
1	OOS	0.00289668	0.02933737	CV	0.00141077	0.02655288
2	OOS	0.00289794	0.02932829	CV	0.00142612	0.02639643
3	OOS	0.00289657	0.02933271	CV	0.00142057	0.02642614
4	OOS	0.00322631	0.03233348	CV	0.00161577	0.02922893
5	OOS	0.00315717	0.03192838	CV	0.00161598	0.02918103
6	OOS	0.00286177	0.02918355	CV	0.00142573	0.02654399
7	OOS	0.00288278	0.02939365	CV	0.00143408	0.02640879
8	OOS	0.00287012	0.02916590	CV	0.00142191	0.02642395
9	OOS	0.00312023	0.03212712	CV	0.00162363	0.02931566
10	OOS	0.00320773	0.03231870	CV	0.00163675	0.02928567
11	OOS	0.00286202	0.02903762	CV	0.00142202	0.02656322
12	OOS	0.00292002	0.02929876	CV	0.00142352	0.02648266
13	OOS	0.00285639	0.02905318	CV	0.00140817	0.02657282
14	OOS	0.00317692	0.03203844	CV	0.00161741	0.02927366
15	OOS	0.00330047	0.03214643	CV	0.00160875	0.02928342
16	OOS	0.00258238	0.03188518	CV	0.00065514	0.02908691
17	OOS	0.00247809	0.03233077	CV	0.00063395	0.02926124
18	OOS	0.00258870	0.03187116	CV	0.00065226	0.02907460
19	OOS	0.00291476	0.03509102	CV	0.00075164	0.03247364
20	OOS	0.00268914	0.03411292	CV	0.00077341	0.03127794
21	OOS	0.00432814	0.03382664	CV	0.00212293	0.03098144
22	OOS	0.00318839	0.03369424	CV	0.00105776	0.03203304
23	OOS	0.00441609	0.03437149	CV	0.00214454	0.03093313
24	OOS	0.00436641	0.03408481	CV	0.00211717	0.03095843
25	OOS	0.00435252	0.03407101	CV	0.00212123	0.03094502
26	OOS	0.00430159	0.03427615	CV	0.00207995	0.03093004
27	OOS	0.00421521	0.03414207	CV	0.00210636	0.03106603
28	OOS	0.00270250	0.03275321	CV	0.00099183	0.03156059

B.4 MDED Model

Exp. No.	Val. Method	Train MSE	Test MSE	Val. Method	Train MSE	Test MSE
0	OOS	0.00291810	0.02938741	CV	0.00141682	0.02656155
1	OOS	0.00289668	0.02933737	CV	0.00141077	0.02655288
2	OOS	0.00289794	0.02932829	CV	0.00142612	0.02639643
3	OOS	0.00289657	0.02933271	CV	0.00142057	0.02642614
4	OOS	0.00322631	0.03233348	CV	0.00161577	0.02922893
5	OOS	0.00315717	0.03192838	CV	0.00161598	0.02918103
6	OOS	0.00286177	0.02918355	CV	0.00142573	0.02654399
7	OOS	0.00288278	0.02939365	CV	0.00143408	0.02640879
8	OOS	0.00287012	0.02916590	CV	0.00142191	0.02642395
9	OOS	0.00312023	0.03212712	CV	0.00162363	0.02931566
10	OOS	0.00320773	0.03231870	CV	0.00163675	0.02928567
11	OOS	0.00286202	0.02903762	CV	0.00142202	0.02656322
12	OOS	0.00292002	0.02929876	CV	0.00142352	0.02648266
13	OOS	0.00285639	0.02905318	CV	0.00140817	0.02657282
14	OOS	0.00317692	0.03203844	CV	0.00161741	0.02927366
15	OOS	0.00330047	0.03214643	CV	0.00160875	0.02928342
16	OOS	0.00258238	0.03188518	CV	0.00065514	0.02908691
17	OOS	0.00247809	0.03233077	CV	0.00063395	0.02926124
18	OOS	0.00258870	0.03187116	CV	0.00065226	0.02907460
19	OOS	0.00291476	0.03509102	CV	0.00075164	0.03247364
20	OOS	0.00268914	0.03411292	CV	0.00077341	0.03127794
21	OOS	0.00432814	0.03382664	CV	0.00212293	0.03098144
22	OOS	0.00318839	0.03369424	CV	0.00105776	0.03203304
23	OOS	0.00441609	0.03437149	CV	0.00214454	0.03093313
24	OOS	0.00436641	0.03408481	CV	0.00211717	0.03095843
25	OOS	0.00435252	0.03407101	CV	0.00212123	0.03094502
26	OOS	0.00430159	0.03427615	CV	0.00207995	0.03093004
27	OOS	0.00421521	0.03414207	CV	0.00210636	0.03106603
28	OOS	0.00270250	0.03275321	CV	0.00099183	0.03156059

B.5 MDARC0 Model

Exp. No.	Val. Method	Train MSE	Test MSE	Val. Method	Train MSE	Test MSE
0	OOS	0.00448266	0.03017307	CV	0.00303947	0.02750740
1	OOS	0.00438130	0.03033511	CV	0.00303238	0.02750745
2	OOS	0.00430287	0.03025093	CV	0.00303029	0.02749531
3	OOS	0.00437446	0.03033042	CV	0.00303387	0.02750327
4	OOS	0.00483698	0.03265952	CV	0.00348099	0.03003066
5	OOS	0.00454929	0.03280596	CV	0.00346311	0.03000364
6	OOS	0.00431780	0.03022475	CV	0.00303372	0.02750280
7	OOS	0.00428226	0.03027964	CV	0.00302512	0.02752506
8	OOS	0.00433264	0.03022978	CV	0.00303425	0.02750700
9	OOS	0.00483722	0.03269294	CV	0.00348099	0.03002765
10	OOS	0.00461842	0.03266099	CV	0.00345385	0.03004614
11	OOS	0.00431539	0.03039954	CV	0.00302905	0.02751775
12	OOS	0.00438233	0.02996020	CV	0.00302115	0.02753189
13	OOS	0.00434922	0.03043683	CV	0.00303007	0.02751388
14	OOS	0.00483704	0.03273492	CV	0.00348273	0.03003572
15	OOS	0.00458588	0.03277171	CV	0.00344387	0.03005103
16	OOS	0.00466319	0.02700114	CV	0.00336784	0.02516014
17	OOS	0.00461577	0.02745185	CV	0.00354831	0.02496900
18	OOS	0.00476953	0.02739168	CV	0.00337998	0.02512452
19	OOS	0.00764765	0.03100629	CV	0.00657263	0.02989996
20	OOS	0.00444478	0.02963286	CV	0.00342736	0.02700091
21	OOS	0.00434652	0.03296846	CV	0.00321992	0.03100073
22	OOS	0.00524396	0.03485069	CV	0.00431465	0.03097653
23	OOS	0.00434673	0.03296139	CV	0.00322381	0.03100515
24	OOS	0.00430099	0.03303431	CV	0.00321900	0.03100902
25	OOS	0.00434598	0.03296466	CV	0.00322306	0.03099847
26	OOS	0.00430347	0.03303308	CV	0.00321791	0.03099947
27	OOS	0.00430900	0.03302431	CV	0.00321454	0.03098931
28	OOS	0.00495135	0.03372132	CV	0.00374306	0.03106068

B.6 MDARC1 Model

Exp. No.	Val. Method	Train MSE	Test MSE	Val. Method	Train MSE	Test MSE
0	OOS	0.00141642	0.06454347	CV	0.00084872	0.06195843
1	OOS	0.00140551	0.06454576	CV	0.00083698	0.06191304
2	OOS	0.00142036	0.06456436	CV	0.00084097	0.06198693
3	OOS	0.00140505	0.06455857	CV	0.00083630	0.06195052
4	OOS	0.00101288	0.07209861	CV	0.00069561	0.07001002
5	OOS	0.00101078	0.07210341	CV	0.00067773	0.07002524
6	OOS	0.00139963	0.06457152	CV	0.00083325	0.06203867
7	OOS	0.00141745	0.06467819	CV	0.00083342	0.06201544
8	OOS	0.00139929	0.06453407	CV	0.00083269	0.06205413
9	OOS	0.00101298	0.07206679	CV	0.00069558	0.06999928
10	OOS	0.00099895	0.07199983	CV	0.00066705	0.07004536
11	OOS	0.00141481	0.06453824	CV	0.00081582	0.06202888
12	OOS	0.00147210	0.06471340	CV	0.00081717	0.06215511
13	OOS	0.00139244	0.06456903	CV	0.00081491	0.06198741
14	OOS	0.00101256	0.07208105	CV	0.00069429	0.07005456
15	OOS	0.00105312	0.07213666	CV	0.00066688	0.06885190
16	OOS	0.00391465	0.20311481	CV	0.00312568	0.19680441
17	OOS	0.00410346	0.20231562	CV	0.00322415	0.19591008
18	OOS	0.00394311	0.20327858	CV	0.00312588	0.19766841
19	OOS	0.00309193	0.23343913	CV	0.00250065	0.22569748
20	OOS	0.00272093	0.22444343	CV	0.00209785	0.21882312
21	OOS	0.00069841	0.10688639	CV	0.00059354	0.10471736
22	OOS	0.00257703	0.36153815	CV	0.00228157	0.35070087
23	OOS	0.00070420	0.10691662	CV	0.00059857	0.10482455
24	OOS	0.00069194	0.10675472	CV	0.00058697	0.10467764
25	OOS	0.00070273	0.10695535	CV	0.00059703	0.10480972
26	OOS	0.00068225	0.10666073	CV	0.00057774	0.10460431
27	OOS	0.00065183	0.10638874	CV	0.00054876	0.10442328
28	OOS	0.00233916	0.34853808	CV	0.00207044	0.33707789

B.7 MKL Model

Exp. No.	Val. Method	Train MSE	Test MSE	Val. Method	Train MSE	Test MSE
0	OOS	0.00317263	0.02519840	CV	0.00172581	0.02505082
1	OOS	0.00303165	0.02636677	CV	0.00171694	0.02569375
2	OOS	0.00308524	0.02636645	CV	0.00174043	0.02583051
3	OOS	0.00303083	0.02638263	CV	0.00170267	0.02582997
4	OOS	0.00296271	0.02759382	CV	0.00194980	0.02781390
5	OOS	0.00311781	0.03039160	CV	0.00181611	0.02960162
6	OOS	0.00284180	0.02667591	CV	0.00163308	0.02560045
7	OOS	0.00285914	0.02701470	CV	0.00166955	0.02577446
8	OOS	0.00289699	0.02647714	CV	0.00162149	0.02558547
9	OOS	0.00352629	0.02869596	CV	0.00182265	0.02870890
10	OOS	0.00388418	0.02896528	CV	0.00165877	0.02879822
11	OOS	0.00369098	0.02642289	CV	0.00163924	0.02600961
12	OOS	0.00356328	0.02658283	CV	0.00171433	0.02627249
13	OOS	0.00369894	0.02653517	CV	0.00159672	0.02591099
14	OOS	0.00310866	0.02870631	CV	0.00179677	0.02871827
15	OOS	0.00372231	0.02985329	CV	0.00164433	0.02945137
16	OOS	0.00455895	0.02952075	CV	0.00285192	0.02704368
17	OOS	0.00452126	0.02911581	CV	0.00306283	0.02785605
18	OOS	0.00460566	0.02918150	CV	0.00284229	0.02729349
19	OOS	0.00395662	0.03142445	CV	0.00257444	0.02886149
20	OOS	0.00331100	0.03116476	CV	0.00180630	0.02869494
21	OOS	0.00492558	0.03211244	CV	0.00241768	0.03347685
22	OOS	0.00360164	0.03824243	CV	0.00241908	0.03478473
23	OOS	0.00437648	0.03288784	CV	0.00253757	0.03325722
24	OOS	0.00494282	0.03280640	CV	0.00241544	0.03278341
25	OOS	0.00405611	0.03280902	CV	0.00241952	0.03268470
26	OOS	0.00511989	0.03361792	CV	0.00241326	0.03418544
27	OOS	0.00462924	0.03433964	CV	0.00227922	0.03362307
28	OOS	0.00333816	0.03611367	CV	0.00219124	0.03538021

B.8 Linear model

Exp. No.	Train MSE	Test MSE
0	4.00E-04	2.92E-01
1	2.09E-29	3.40E-01
2	9.90E-30	3.11E-01
3	2.44E-29	3.65E-01
4	1.61E-26	7.99E-01
5	9.91E-30	3.30E-01
6	1.04E-29	3.26E-01
7	6.30E-30	2.98E-01
8	1.66E-29	3.28E-01
9	4.21E-27	5.00E-01
10	4.21E-30	3.21E-01
11	4.29E-30	3.20E-01
12	3.77E-30	2.93E-01
13	4.18E-30	3.22E-01
14	2.77E-27	4.51E-01
15	2.04E-30	3.14E-01
16	5.30E-30	1.08E-01
17	4.02E-30	1.12E-01
18	4.02E-30	1.24E-01
19	2.93E-27	2.82E-01
20	2.01E-30	9.96E-02
21	9.90E-30	4.60E-01
22	6.73E-30	1.71E-01
23	4.33E-29	4.94E-01
24	6.02E-30	4.59E-01
25	3.22E-29	4.70E-01
26	3.45E-30	4.47E-01
27	1.83E-30	4.35E-01
28	1.84E-30	1.34E-01